Federal Office
for Information Security

Quantum Machine Learning in the Context of IT Security

Security within Quantum Machine Learning &
Quantum Machine Learning for Cyber Security

Abstract

Since the training of modern learning systems is a time consuming endeavour, the idea of leveraging quantum computing for solving machine learning problems becomes increasingly popular. Indeed, as first quantum machine learning algorithms have been successfully implemented on present day quantum devices, it seems reasonable to expect that machine learning routines will be executed on reliable, large scale quantum computers once such devices become available. However, instead of only focusing on benefits due to quantum parallelism or quantum encoding, it seems prudent to also investigate potential quantum machine learning risks with respect to reliability, trustworthiness, safety, security, and maleficent use.

In this study, we therefore ask how quantum computing may impact machine learning and whether quantum machine learning may serve as a tool in cyber defense or cyber offence. To this end, we first summarize fundamental concepts in quantum computing and classical machine learning and provide an overview over current developments in the nascent field of quantum machine learning. Then, we assess whether quantum machine learning solutions may be more secure than classical ones. To this end, we look at attack surfaces in quantum computing and provide classification dimensions for possible types of attacks against and possible defense strategies for quantum machine learning. Finally, we investigate quantum machine learning methods as possible tools in cyber security. To this end, we investigate their potential application by attackers and defenders alike and discuss several specific practical application scenarios such as spam-, malware- or attack detection.

Our results are based on an extensive survey of the relevant scientific literature. We carefully consider what is technically feasible on present day quantum devices and critically scrutinize scientific publications on quantum machine learning and quantum IT security with respect to the technical feasibility of what they report.

Due to the novelty of the field of quantum machine learning and remaining technical uncertainties regarding its future deployment in practice, definitive answers regarding potential (in)vulnerabilities of quantum machine learning and its potential role in cyber security are not always possible. We therefore suggest several directions for further investigation of quantum machine learning in the context of IT security. These address questions such as “How to quantify the robustness of quantum machine learning against noise or adversarial attacks?”, “Is quantum machine learning robust against model stealing?”, or “Does quantum machine learning allow for easier generation of fakes or deceit of security systems?” and may inform future efforts. A positive consequence of the remaining uncertainties as to the future practical applicability of quantum machine learning, however, is the opportunity for IT security researchers, developers, and policy maker to stay ahead of the curve. At the current development stage, it still seems possible to rally relevant stakeholders to begin to fathom necessary and appropriate standardization measures or guidelines for development and deployment of quantum machine learning solutions.

Authors

Christian Bauckhage, Fraunhofer IAIS
Rainer Bye, Capgemini
Christian Knopf, Capgemini
Merima Mustafic, Capgemini
Nico Piatkowski, Fraunhofer IAIS
Barry Reese, Capgemini
Roland Stahl, Capgemini
Eldar Sultanow, Capgemini

Publisher

Federal Office for Information Security
`https://www.bsi.bund.de`

Contents

1 Introduction and Motivation 5

Part I: Quantum Machine Learning 9

2 Introduction to Quantum Machine Learning 11

3 Quantum Computing in a Nutshell 15
3.1 Adiabatic Quantum Computing . . . 20
3.2 Quantum Gate Computing . . . 24
3.3 Further Quantum Computing Paradigms . . . 28
3.3.1 Topological Quantum Computing . . . 28
3.3.2 Variational Quantum Computing . . . 29
3.4 Technical State of the Art and Limitations . . . 29

4 Machine Learning in a Nutshell 31
4.1 The Machine Learning Pipeline . . . 32
4.2 Lazy vs. Eager Machine Learning . . . 36
4.3 Sources of Uncertainty in Machine Learning . . . 36

5 Quantum Machine Learning 39
5.1 Quantum Inspired Machine Learning . . . 39
5.1.1 Tang's Quantum Inspired Algorithm for Recommendation Systems . . . 41
5.1.2 Tensor Networks . . . 42
5.1.3 Digital Annealing . . . 44
5.1.4 Quantum Inspired Data Clustering . . . 46
5.1.5 Quantum Inspired Gravitational Search . . . 46
5.2 Machine Learning for Quantum Computing . . . 47
5.2.1 Quantum Circuit Design . . . 47
5.2.2 Quantum State Preparation . . . 48
5.2.3 Quantum Noise Modelling . . . 49
5.3 Quantum Enhanced Machine Learning . . . 50
5.3.1 Quantum Algorithms for Linear Algebra . . . 51
5.3.2 Quantum Algorithms for Regression . . . 54
5.3.3 Quantum Algorithms for Clustering . . . 56
5.3.4 Quantum Algorithms for Nearest Neighbor Search . . . 57
5.3.5 Quantum Algorithms for Classification . . . 59
5.3.6 Quantum Boosting . . . 62
5.3.7 Quantum Support Vector Machines . . . 63
5.3.8 Quantum Neural Networks . . . 65
5.3.9 Federated and Distributed Quantum Machine Learning . . . 70
5.3.10 Variational Quantum Algorithms . . . 71

5.4 Quantum Machine Learning for Quantum Data 75

6 Characteristics of QML Methods 77

6.1 Hardware Requirements 79

6.1.1 Quantum Memory 79

6.1.2 Qubit Connectivity 81

6.1.3 Qubit Count 82

6.1.4 Quantum Circuit Depth 84

6.2 Algorithmic Requirements 85

6.2.1 Data Pre- and Post-Processing 85

6.2.2 Statistical Error 86

6.2.3 Noise Robustness 88

7 Conclusion and Outlook of Part I 90

Glossary for Part I 99

Part II: Security within Quantum Machine Learning 109

8 Security in the Application of QML 111

9 The Quantum Machine Learning Pipeline 115

9.1 The Software Engineering Lifecycle in Quantum Computing 115

9.2 Attack Surfaces in Quantum Computing 117

9.3 The ML/QML Pipeline 118

9.4 The QML Pipeline in the Hybrid Setting 119

9.5 Attack Surfaces in the QML Pipeline 122

9.6 Utilization of Quantum Computing for the Training Phase 125

10 Classification Dimensions for Attack Types 127

10.1 Influence Capabilities of the Attacker 127

10.2 Attacked Property of the System 127

10.3 Attack Specificity 128

10.4 Attack Types 128

11 Attacks on QML Systems 132

11.1 Model Stealing Attacks 132

11.2 Adversarial Attacks 134

11.3 Data Poisoning Attacks 138

11.4 Privacy Attacks 142

11.5 Side-Channel Attacks 144

11.6 Other Attacks 148

12 Additional Defense Strategies **151**
12.1 Defending Attacks on Privacy 152
12.2 Robustness and Noise of QNNs 154
12.3 Defenses of QML Based on ML 155

13 Conclusion and Outlook of Part II **157**

Part III: Quantum Machine Learning in Cyber Security **161**

14 QML as a Tool in Cybersecurity **163**
14.1 Information and IT-Security 164
14.2 Attacker and Defender 165
14.3 Machine Learning in Cyber Security 167

15 Spam Detection **169**
15.1 Definition and Current Research on Spam 170
15.2 The Underlying Approach: Text Classification 171

16 Malware Detection **173**
16.1 Historical Signature-Based Approaches 173
16.2 ML-Based Approaches 173
16.3 QML Approaches and Current Research 175
16.4 Detection of Malware in Quantum Computers 178

17 Attack Detection **180**
17.1 ML for Attack Detection 180
17.2 QML Approaches and Results 180
17.3 Advanced Detection 181
17.4 Quantum Algorithms for Quantum Data 182

18 Overarching Considerations **183**
18.1 Measuring, Comparing, and Optimizing (Q)ML Models 183
18.2 Model Decay 184
18.3 Synthetic Training Data 184
18.4 Robustness and Expressability in QML 185
18.5 Adversarial Attacks in Cyber Security 186
18.6 A Case for Deep Learning in Cyber Security QML 188
18.7 An Attacker with Dominant Quantum Capabilities 188

19 Conclusion and Outlook of Part III **191**

Literature **195**

1 Introduction and Motivation

This study addresses *quantum machine learning* and its potential role in the context of *IT security*. Given this topic, it seems warranted to first of all answer two arguably obvious questions, namely "why would this topic be of current interest?" and "why would it merit an in-depth scientific assessment at this point in time?" In order to motivate the content of this study, we will therefore begin by taking stock of current general trends in IT.

The ever accelerating digitization of modern societies is a key technological development in the early 21st century. The Internet, a plethora of web-based platforms, widely available mobile communication devices, and increasingly computerized machinery are fundamentally transforming entertainment and social interaction, banking and business, shopping and transport, production and logistics, medicine and healthcare, as well as administration and government. All of this impacts our private and professional lives, facilitates many processes, and increases their efficiency or sustainability. At the same time, this transformative period is characterized by increased risks with respect to privacy and information security. Cybercrime and -warfare threaten individuals, companies, organizations, and (critical) infrastructures. Malware, ransomware, and data theft cause billions of dollars of damage, can paralyze businesses or administrations, and ruin lives. Indeed, Cybersecurity Ventures recently estimated that global cybercrime has become of the largest money movers worldwide and that crime as a service and ransomware have become- and will likely remain "growth industries". This is of considerable concern as cybercrime threatens innovation as well as investment and is often quoted as more profitable than the global trade of all major illicit drugs combined.

While cybercrime is not a novel phenomenon, its recent proliferation is. On one hand, its steady rise is of course but an immediate consequence of the increased widespread use of digital technologies. On the other hand, however, IT security now also faces genuinely novel challenges due to the increased deployment of *artificial intelligence* for process automation.

The idea of artificial intelligence (AI) is not new either but can actually be traced back to the early days of electronic computing. As early as in the 1940s, scientists began thinking about "artificial brains" and concepts such as artificial neurons, i.e. information processing circuits that mimic the functionality of biological neurons, date back to pioneering work by McCulloch and Pitts in 1943. What is novel about AI at the beginning of the 21st century is that, after decades of worldwide research and development, it has now reached a level of technical maturity that encourages its deployment in practice.

To understand why this is remarkable, we note that there exist two main branches of AI, namely symbolic- and subsymbolic or, equivalently, knowledge- and data-driven systems. While both have been around for decades, AI research in the 20th century was largely dominated by the former. Broadly speaking, knowledge-driven AI systems are computer implementations of human knowledge or ingeniously designed procedures. Examples include expert systems, tree-search algorithms, or specifically engineered signal processing methods. Exemplary achievements within this paradigm include the Deep Blue chess computer which, in 1997, was able to beat the reigning world champion, as well as Dickmann's self-driving cars VaMP and VITA-2 which were able to drive autonomously on the french Autoroute 1 in 1994.

Data-driven AI systems, on the other hand, are largely problem agnostic and acquire their problem solving capabilities by means of *machine learning*. The idea of machine learning (ML) has a venerable history, too. Indeed, the term was already coined by Turing's 1948 paper "Intelligent Machinery" in which he considered the idea of training networks of artificial neurons to perform cognitive tasks.

The basic idea behind constructing ML systems is to consider very general parameterized mathematical models that can be implemented on a computer and then be adapted to a given task. To this end, one considers large sets of problem-specific data which are analyzed by a learning algorithm which adjusts the model parameters such that the trained model shows the intended or desired input-output behavior. ML systems have traditionally been used for solving pattern recognition problems such as image understanding or speech recognition. Over the past decade, however, researchers generalized them to a much wider variety of AI problems so that they can now play games, conduct natural language dialogues, plan complex tasks, or simulate physical processes. Major drivers behind this recent progress in data-driven AI are the abundance of data in our digitized world and ever more powerful computers which allow for implementing and training very large mathematical models.

In fact, at present, the notion of general parameterized mathematical models is basically synonymous with (deep) artificial neural networks, which, when designed and trained properly, do indeed achieve remarkable performance. However, when compared to traditional knowledge-driven AI approaches, they come along with two major issues that are of interest four this study.

First of all, modern neural networks involve millions if not billions of adjustable parameters and thus constitute highly complex systems. That is to say that trained neural networks are largely black boxes whose input/output behavior can be observed and evaluated but whose inner workings are opaque even to experts. Moreover, depending on the data such networks have been trained with, their decisions can be biased or they may be susceptible to data poisoning leading to failures upon adversarial inputs. From the point of view of IT security, this raises questions as to the vulnerability, reliability, and traceability of modern learning systems.

Second of all, the training of modern learning systems is a compute- and data intensive process with considerable resource demands. Indeed, the computations required to adjust billions of model parameters to a problem at hand are so demanding that the training of modern deep learning systems typically necessitates special purpose hardware (i.e. high-performance GPU clusters, or even application specific circuits like TPUs) and may even then take weeks until completion. This now motivates a quickly growing number of researchers to look for alternative training procedures. An ever more popular idea in this regard is to try to harness *quantum computing* for machine learning.

Initial ideas for how to harness quantum mechanics for information processing, i.e. for *quantum computing* (QC), can be traced back to the early 1980s when pioneers such as Feynman and Manin realized that, at least for certain problems, quantum computers should be able to outperform digital computers. Indeed, shortly after Deutsch and Jozsa, Shor, and Grover reported quantum algorithms (for rather academic- as well as for important practical problems) which showed quantum advantages in form of accelerated computations due to quantum parallelism. Other potential quantum advantages of interest for machine learning include higher information density due to quantum entanglement or inherently higher security against outside interference due to quantum decoherence. However, it took until the early 2000s until working quantum computers on which to run such algorithms became technical reality. Ever since, the development of quantum computers has accelerated noticeably as well, and thus constitutes yet another current IT trend of highly disruptive potential.

Given that reasonably powerful quantum computers have now become commercially available, it is hardly a surprise that efforts on *quantum machine learning* (QML) are increasing noticeably. Against this backdrop, it therefore now seems opportune to ask if and how quantum computing may impact machine learning and whether quantum machine learning solutions may serve as tools in cyber defense or cyber offence.

This study attempts to answer these questions by means of carefully surveying the scientific literature on QC and QML. It is structured into the following three major parts:

- Part I: **Quantum Machine Learning**,
- Part II: **Security within Quantum Machine Learning**, and
- Part III: **Quantum Machine Learning in Cyber Security**.

Part I provides the technical background and is intended to clarify the notion of quantum machine learning. There, we first recall the basic ideas and fundamental concepts of quantum computing. We then discuss theory and practice of classical machine learning, and finally give an account of the current state of the art and of current algorithms and approaches in the nascent field of quantum machine learning.

In part II, we then address the question of whether or not quantum machine learning solutions may be more secure than classical systems. To this end, we discuss the quantum machine learning pipeline, look at attack surfaces in quantum computing, and provide classification dimensions for possible types of attacks against- and possible defense strategies for quantum machine learning.

In part III, we finally consider quantum machine learning methods as possible tools in cyber security. To this end, we investigate their potential application by attackers and defenders alike and discuss several specific practical use cases such as malware- or attack detection.

Our discussion in parts II and III will largely assume the point of view of currently existing quantum computing hardware. That is, we will consider the kind of capabilities current quantum computing devices offer for the realization of quantum machine learning solutions. In other words, we will critically scrutinize the existing scientific literature on quantum machine learning and quantum IT security with respect to the technical feasibility of what is reported. This critical approach appears to be warranted since existing quantum computers are not yet widely accessible and, importantly, still very much limited with respect to their capabilities. Indeed, lack of accessibility and, as a consequence, lack of testability may be reasons for why much of the ongoing research on quantum machine learning algorithms posits the existence of universal quantum computers with fully realized quantum advantages. However, such devices do not yet exist. Hence, not every reported quantum computing idea is of current practical use and thus of limited concern with respect to near-term security considerations. Put differently, while quantum computing hardware is currently evolving rapidly, the physical or technical realization of truly universal quantum computers still poses many difficult engineering problems and, as of this writing, it is not yet clear if these can be overcome or not.

Due to the overall novelty of the field of quantum machine learning and due to the remaining uncertainty regarding the development of fully fledged quantum computers which practically realize all the features theory predicts, the scientific assessments in parts II and III will necessarily point to many open questions. In other words, definitive answers regarding potential (in)vulnerabilities of quantum machine learning and its potential role in cyber security are as of now not always possible. Throughout the text, we therefore highlight identified open questions and use them as a basis for finally suggesting further specific or targeted research on quantum computing in the context of cyber security.

Part I

Quantum Machine Learning

2 Introduction to Quantum Machine Learning

One of the most striking and likely most disruptive technological developments in the early 21st century is the maturing of *Artificial Intelligence* (AI). After decades of worldwide research which began in the 1940s, (weakly) intelligent technical systems have now become reality and are increasingly deployed in practice. Prominent examples of recent accomplishments include cognitive systems for robust text- and speech recognition and generation [1, 2, 3], image- and video understanding and synthesis [4, 5, 6], game play [7, 8], recommendation [9, 10], (medical) diagnostics [11, 12], planning and decision making [13, 14, 15], and convincing conversational agents [16, 17, 18].

Achievements like these are mainly due to progress in *Machine Learning* (ML), a branch of AI concerned with adjusting the parameters of a software system in a training process such that it can develop cognitive capabilities. In other words, while there were many innovations in many branches of AI, the recent performance boost is due to systems which learn an intended input-output behavior from analyzing task specific example data [19, 20, 21, 22].

In and off itself, machine learning is not a new idea; it was initially conceptualized by Turing and had its first boom period in the 1980s after algorithms for training artificial neural networks became available. Its accelerated improvement over the past decade can be attributed to four technological trends:

1. exponentially growing amounts of (training) data available on the Internet, mainly due to social media platforms and mobile communication devices, but in no small parts also due to the digitization of industry and the life- and natural sciences

2. increasing capabilities and decreasing costs of high performance computing hardware, most notably the availability of GPU computing for serious applications

3. ever more rapidly developed and publicly disseminated open source software and frameworks for the design, training, and deployment of machine learning systems

4. last but not least, scientific progress in neurocomputing, most notably in the areas of deep neural networks and deep learning [23]

Indeed, a conclusion to be drawn from the past decade of machine learning research is that these trends had to come together to enable results such as mentioned above. Although there exist solutions for very capable learning systems of comparatively moderate sizes [24, 25], the capability of a learning system to exhibit human-like performance seems to be predicated on the fact that its size far exceeds traditional system sizes. In other words, the underlying mathematical models seem to require more flexibility and thus more adjustable parameters than have been considered historically.

For instance, the adjustable parameters of an artificial neural network are its synaptic weights and modern deep networks often come with hundreds of billions of such weights (see e.g. [18]). In order for such flexible models to reliably learn a desired cognitive skill, classical results in learning theory suggest that they must be trained with large amounts of representative training data [26]. Indeed, state of the art systems are commonly trained with up to several billion data points (see e.g. [18]). Given numbers like these, it is clear that the training of modern learning systems is a resource intensive endeavor that easily translates to several hundred GPU years of computation time (see e.g. [18]). Modern machine learning has therefore reached a point where practical feasibility and success are conditioned on access to high performance computing hardware.

Against this backdrop, it is not surprising that machine learning researchers are beginning to look at another technology that has recently seen substantial progress, namely *quantum information processing* (QIP) or *quantum computing* (QC).

First ideas for quantum information processing were published in the 1970s [27, 28, 29, 30] and the theoretical foundations of quantum computing date back to the 1980s when Russian and American physicists first conceptualized quantum computers [31, 32, 33, 34]. The appeal of these early contributions was that they showed that quantum bits can carry more information than classical bits so that quantum computing could be expected to solve certain difficult problems much faster than classically possible.

It then took about another decade until first quantum algorithms were conceived which, when running on a quantum computer, would exhibit these hypothesized advantages [35, 36, 37]. However, at the time these algorithms were presented they were mere theoretical exercises in “pen and paper programming” since quantum computers on which they could have been implemented did not yet exist. Nevertheless, Grover’s amplitude amplification algorithm convincingly demonstrated that quantum computers can exhaustively search unstructured lists quadratically faster than digital computers [37, 38] and Shor’s celebrated quantum algorithm for large integer factorization even revealed exponential speedup over classical approaches [36, 39]. The latter sparked broader attention to the idea of quantum computing, because Shor’s discovery implied that common data encryption schemes would no longer be secure if working quantum computers were to become technical reality. In short, these pioneering contributions therefore demonstrated that quantum algorithms running on ideal quantum computers would indeed exhibit *quantum supremacy* when dealing with certain difficult problems. They also made clear that this quantum supremacy would have considerable practical implications.

First prototypes of genuine quantum computers appeared in the late 1990s. They were genuine in that they were made up of quantum hardware for physical quantum information processing as opposed to digital hardware for simulated quantum information processing. For example, physical implementations of the Deutsch-Jozsa algorithm [35] were reported in 1998 [40, 41] and Shor’s algorithm was first physically implemented by a team at IBM in 2001 [42]. Since then, technical developments progressed quickly and have meanwhile led to commercially accessible quantum computing platforms developed by various companies.

This state of affairs and the ever intensifying efforts towards marketed solutions confirm that quantum computing really is a viable idea; proof of concepts exist, claims as to first practical observations of quantum supremacy have been made [43], and a growing number of practitioners see remaining challenges in the development of industrial strength quantum computers as mere engineering problems rather than as fundamental research problems.

It therefore seems likely that *quantum machine learning* (QML) will become practical, too, and that quantum information processing will become applicable at various stages of the machine learning pipeline.

Indeed, since many of the fundamental problems in machine learning are demanding optimization or search problems of the kind quantum computing can excel at, research on QML is noticeably intensifying. The corresponding literature has seen rapid growth and first textbooks and tutorials have become available [44, 45, 46, 47, 48]. Moreover, more and more quantum coding challenges are being held that specifically focus on aspects of machine learning and there are recognizable efforts by industrial players to get machine learners involved in quantum computing research.

Given these developments, experts predict that quantum machine learning on reliable, large scale quantum computers will soon facilitate demanding learning tasks and even put problems into reach which are intractable on digital computers. Should these predictions materialize, QML technology will further accelerate progress in AI which, in turn, will likely lead to novel marketable products and commercial solutions impacting economies and societies [49, 50].

However, next to its potential benefits, QML may also come with certain risks. Since learning systems now see commercialization and industrial deployment even in sensitive areas such as autonomous driving or financial services [51, 52, 53, 54], questions as to their ethics, reliability, trustworthiness, and safety are becoming more urgent than when machine learning was a mere academic endeavour [55, 56, 57, 58]. In fact, now that machine learning and quantum computing are beginning to coalesce, these questions are likely to become ever more important. In other words, given the expected impact of quantum machine learning on capabilities and utilizability of artificial cognitive systems, it seems appropriate to assess potential security issues related to quantum machine learning.

In preparation of such an assessment, this present report is intended to take stock and to give an overview of the current state of the art in quantum machine learning. In section 3, we first provide a short introduction to quantum computing, its basic principles, and major paradigms. Next, in section 4, we clarify the notion of machine learning and elaborate on common settings and methods.

Section 5 then surveys and reviews the scientific literature on quantum machine learning. There, we will adhere to the currently established QML taxonomy and consider quantum inspired classical algorithms for classical data, genuine quantum algorithms for classical data, classical algorithms for quantum data, and quantum algorithms for quantum data. Our presentation with respect to genuine quantum algorithms will assume a point of view where we abstract from physical aspects of the hardware of quantum computers. Rather, we will follow common practice and focus on logical aspects or higher level abstractions that programmers and software developers are typically exposed to.

Having said this, it is nevertheless important to acknowledge that, as of this writing, quantum algorithm design is still very much a theoretic endeavor and that many theoretically valid ideas cannot be implemented on present day quantum computing devices. In other words, physical or hardware related aspects have to be kept in mind when assessing the validity of proposed quantum machine learning solutions. To this end, section 6 will review common assumptions as to data pre- and post-processing, robustness against measurement noise, numbers of available quantum bits, or required quantum circuit depths and will also assess how realistic such assumptions are and what they mean for the practical feasibility of currently proposed QML solutions.

Finally, in section 7, we will summarize key points of this report and provide an outlook to planned studies on quantum machine learning security.

3 Quantum Computing in a Nutshell

Quantum computing exploits quantum mechanical phenomena for information processing. This offers great computational power but also requires a different kind of thinking than in the domain of digital computing. The latter may explain why quantum computing has still not yet received widespread attention in mainstream computer science. As of this writing, major obstacles for a broader engagement with the topic likely are as follows: 1) quantum mechanical phenomena such as superposition, entanglement, tunneling, decoherence, or the uncertainty principle appear to be weird, abstract or unintuitive, and hard to accept for they cannot be observed in our daily macroscopic environments; 2) the mathematical tools used to model these phenomena are complex (no pun intended) and again rather abstract; and 3) to the uninitiated, the mathematical notation used in the quantum computing literature appears even more abstract and needs getting used to. Acknowledging these difficulties, this section provides a brief introduction to the basic terminology of- and fundamental concepts behind quantum computing.

To begin with, we recall that a *quantum mechanical system* (QM system) is characterized by a state vector $|\psi\rangle$ in a Hilbert space $\mathbb{H}$ over $\mathbb{C}$. The temporal evolution of such a system is governed by the Schrödinger equation and its observable physical properties (such as position, momentum, or energy) correspond to Hermitian operators. A measurement of an observable property of a QM system collapses its state to an eigenvector of the respective operator. This is to say that a measurement of a quantum variable results in an eigenvalue of the operator under consideration and the state "jumps" to the corresponding eigenvector. Which of the eigenvalues and corresponding eigenvectors this will be depends on certain probabilities encoded in the state vector. The crucial question as to why this kind of mathematics describes the behavior of nature on one of its most fundamental levels still awaits conclusive answers; for now, the empirically undeniable success of this linear algebraic framework has to be attributed to "the unreasonable effectiveness of mathematics in the natural sciences" [59].

Quantum computing (QC) deals with quantum mechanical systems that represent *quantum bits* or *qubits*. All throughout our following discussion, we will focus on *logical qubits* which are the abstract basic building blocks on which quantum algorithms operate. A *physical qubit*, on the other hand, is a physical realization of a logical qubit, namely a physical device that behaves like a logical qubit and forms a component of the hardware of a quantum computing system. We will neither discuss possible technologies for building such devices, nor will we discuss the practical advantages and disadvantages of different such technologies. An extensive overview of the state of the art in this area can be found in a recent study published by the Federal Office for Information Security [60].

From the point of view of quantum algorithms, our focus on logical qubits makes sense. It mimics common levels of abstraction in modern software development where programmers usually do not have to pay attention to hardware details. In fact, the distinction between hard- and software aspects of computing was essential for the success of digital computers once they matured. Current quantum computers, however, have not yet reached this level of maturity. They are *noisy intermediate-scale quantum* (NISQ) devices [61] which contain less than a hundred physical qubits and still suffer from limited coherence times and low fault-tolerance. From the point of view of existing quantum hardware, logical qubits are therefore still an idealization since they abstract away technological shortcomings. This may render some of the mathematically valid ideas for quantum algorithms practically infeasible and, as of this writing, constitutes yet another difference between quantum- and digital computing. We will return to this issue in section 6.

On a classical computer, the basic units of information are bits and the mathematics that describes their behavior is Boolean algebra. On a quantum computer, the basic units of information are qubits and the mathematics that models their behavior is complex linear algebra.

While a classical bit is always in one and only one of two possible states (0 or 1), the basic tenet of quantum information processing is that a qubit exists in a *superposition* of two *basis states* and collapses to either one once measured. Examples of well known physical *two-state quantum systems* that exhibit this phenomenon include the polarization of a photon, the spin of an electron, or the ground- and first excited state of an atom. Yet, it is interesting to note that quantum mechanical superposition is not exclusive to the subatomic world but may just as well occur in completely isolated macroscopic systems. This is, for instance, used in quantum information processing devices made of superconducting circuits in which electrical currents flow in two directions simultaneously.

Superposition is crucial because it implies that the state space of a qubit is not confined to only two states. It rather consists of infinitely many states which can be represented as two-dimensional, complex-valued unit vectors that are linear combinations of two distinguished, linearly independent, orthonormal basis states. Using the Dirac notation, these basis states are commonly written as $|0\rangle$ and $|1\rangle$ and typically thought of as

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \qquad \text{and} \qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{1}$$

In terms of a mathematical equation, the above translates to the statement that (the state vector of) a qubit can be written as

$$|\psi\rangle = \alpha_0\,|0\rangle + \alpha_1\,|1\rangle \tag{2}$$

where the coefficients $\alpha_0, \alpha_1 \in \mathbb{C}$ are called the *amplitudes* of the basis states $|0\rangle$ and $|1\rangle$, respectively. Importantly, these amplitudes have to obey the normalization condition

$$|\alpha_0|^2 + |\alpha_1|^2 = 1 \tag{3}$$

and are interpreted as follows: if a measurement is performed on qubit $|\psi\rangle$, the probability of finding it in basis state $|0\rangle$ is $|\alpha_0|^2$ whereas the probability of finding it in basis state $|1\rangle$

corresponds to $|\alpha_1|^2$. In other words, the probability of measuring the qubit in, for instance, state $|0\rangle$ is given by $|\langle 0|\psi\rangle|^2$, because the inner product of $|0\rangle$ and $|\psi\rangle$ evaluates to

$$\langle 0|\psi\rangle = \alpha_0\langle 0|0\rangle + \alpha_1\langle 0|1\rangle = \alpha_0 \cdot 1 + \alpha_1 \cdot 0 = \alpha_0 \tag{4}$$

Note that measurements performed on $|\psi\rangle$ are irreversible operations because they constitute interactions with the outside world and therefore lead to loss of superposition or quantum *decoherence*. In other words, once a qubit has collapsed to either one of its basis states, it henceforth behaves like a classical bit.

Operations on qubits which preserve their quantum mechanical nature are called *reversible*. Mathematically, these correspond to unitary linear operators $U \in SU_2(\mathbb{C})$ for which $UU^\dagger = U^\dagger U = I$. Reversible operators can also be written as $U = e^{-iHt/\hbar}$ where H is yet another operator called the Hamiltonian. It corresponds to the total energy of a quantum system in the sense that its spectrum is the set of possible outcomes of measurements of the system's total energy.

Another tenet of quantum information processing is that qubits can be combined to form qubit registers. While a single qubit $|\psi\rangle$ exists in a superposition of 2 states, a quantum register $|\boldsymbol{\psi}\rangle$ of n qubits exists in a superposition of 2^n states. Mathematically, this is to say that

$$|\boldsymbol{\psi}\rangle = \sum_{i=0}^{2^n-1} \gamma_i \, |\boldsymbol{\psi}_i\rangle \tag{5}$$

Here, the γ_i once again obey $\sum_i |\gamma_i|^2 = 1$ and the $|\boldsymbol{\psi}_i\rangle$ are 2^n-dimensional tensor products of single qubit states. Mathematically, these tensor products are Kronecker products and behave as follows: If we consider two single qubits represented as

$$|\psi\rangle = \alpha_0\,|0\rangle + \alpha_1\,|1\rangle = \begin{bmatrix}\alpha_0\\ \alpha_1\end{bmatrix} \qquad \text{and} \qquad |\phi\rangle = \beta_0\,|0\rangle + \beta_1\,|1\rangle = \begin{bmatrix}\beta_0\\ \beta_1\end{bmatrix} \tag{6}$$

then the state of a system composed of these two qubits is given by the four-dimensional vector

$$|\psi\rangle\otimes|\phi\rangle = \begin{bmatrix}\alpha_0\\ \alpha_1\end{bmatrix}\otimes\begin{bmatrix}\beta_0\\ \beta_1\end{bmatrix} = \begin{bmatrix}\alpha_0\beta_0\\ \alpha_0\beta_1\\ \alpha_1\beta_0\\ \alpha_1\beta_1\end{bmatrix} = \alpha_0\beta_0|0\rangle\otimes|0\rangle+\alpha_0\beta_1|0\rangle\otimes|1\rangle+\alpha_1\beta_0|1\rangle\otimes|0\rangle+\alpha_1\beta_1|1\rangle\otimes|1\rangle \tag{7}$$

This example illustrates that the 2^n dimensional state space of an n qubit system is spanned by a *computational basis* that consists of 2^n basis vectors which are tensor products of individual basis states. Since such tensor products over basis states occur in many quantum computing equations, they are typically written more succinctly. For instance, for a quantum register where $n = 3$, we would write the $2^3 = 8$ basis states as

$$|\psi_0\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \equiv |000\rangle \tag{8}$$
$$|\psi_1\rangle = |0\rangle \otimes |0\rangle \otimes |1\rangle \equiv |001\rangle \tag{9}$$
$$\vdots$$
$$|\psi_7\rangle = |1\rangle \otimes |1\rangle \otimes |1\rangle \equiv |111\rangle \tag{10}$$

or, even shorter as $|i\rangle$ where $i \in \{0, \ldots, 7\}$. Since the state space of an n qubit system is 2^n-dimensional, operators acting on such states are (products of) $2^n \times 2^n$ dimensional unitary matrices. Yet another tenet of quantum information processing is that qubits can be entangled. The physical phenomenon of entanglement is indeed an essential aspect of quantum information that has no immediate classical counterpart. To understand its characteristics, we note that the 2 qubit system in (7) has a complete state that is a tensor product of individual qubit states. However, systems of qubits can also be in states that can not be expressed in terms of tensor products. These are called *entangled states* and a canonical example of such a state is

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{11}$$

The following simple computations will illustrate why entangled states are special: Consider two systems $|\psi_1\rangle$ and $|\psi_2\rangle$ of 2 qubits where $|\psi_1\rangle$ is in a tensor product state

$$|\psi_1\rangle = \gamma_0|00\rangle + \gamma_1|01\rangle + \gamma_2|10\rangle + \gamma_3|11\rangle \tag{12}$$

and $|\psi_2\rangle$ is in the entangled state

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{13}$$

If we now perform a partial measurement of both systems where only the first qubit of either system is measured and the second one is left intact, the probability of finding the first qubit of system $|\psi_1\rangle$ in, say, state $|0\rangle$ is given by

$$\sum_{x\in\{0,1\}} |\langle 0x|\psi_1\rangle|^2 = |\gamma_0|^2 + |\gamma_1|^2 \tag{14}$$

Moreover, once this partial measurement has been performed, the system will be in the new (re-normalized) state

$$|\hat{\psi}_1\rangle = \frac{\sum_{x\in\{0,1\}}\langle 0x|\psi_1\rangle\,|0x\rangle}{\sqrt{\sum_{x\in\{0,1\}}|\langle 0x|\psi_1\rangle|^2}} = \frac{\gamma_0|00\rangle + \gamma_1|01\rangle}{\sqrt{|\gamma_0|^2 + |\gamma_1|^2}} \tag{15}$$

In other words, measuring the first qubit of the first system will cause this system to still be in a superposition of two computational basis states. In our example, this means that the state of the second qubit remains undetermined since it could still be found either in state $|0\rangle$ or in state $|1\rangle$.

If, on the other hand, the same kind of partial measurement is performed on the second system $|\psi_2\rangle$, the probability of finding its first qubit in state $|0\rangle$ is given by

$$\sum_{x\in\{0,1\}} |\langle 0x|\psi_2\rangle|^2 = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} \tag{16}$$

and, upon this measurement, the second system will be in state

$$|\hat{\psi_2}\rangle = \frac{\sum_{x\in\{0,1\}} \langle 0x|\psi_2\rangle\, |0x\rangle}{\sqrt{\sum_{x\in\{0,1\}} |\langle 0x|\psi_2\rangle|^2}} = |00\rangle \tag{17}$$

By the same token, if we had found the first qubit in state $|1\rangle$, the final state of the system would have been $|\hat{\psi_2}\rangle = |11\rangle$ and similar outcomes would have been observed if we had measured the second qubit of the entangled system. That is, whenever two qubits are entangled, their individual states cannot be measured separately; a measurement of either one of two entangled qubits also determines the state of the other.

Entanglement is of vital importance for quantum information processing because it is this phenomenon which makes it possible to consider only n (physical) qubits to perform computation in a 2^n dimensional state space. In other words, whereas doubling the "processing power" of a digital computer would require doubling the number of bits, doubling the "processing power" of a quantum computer only requires adding one extra qubit. Without entanglement, this exponential increase in performance is provably impossible [62].

Given these prerequisites, we can now discuss the two major paradigms for practical quantum computing, namely quantum gate computing and adiabatic quantum computing. The former approaches problem solving by sequencing unitary operators, $U_1, U_2, \ldots$ into quantum circuits and well known examples include Grover's search algorithm [38] or Shor's integer factorization algorithm [39]. The latter considers problem solving as an energy minimization task and is concerned with finding ground states of problem specific Hamiltonian operators H. Adiabatic quantum computing is sometimes falsely said to be less general than quantum gate computing but both paradigms are provably equivalent with respect to computational power or expressiveness [63, 64]). Indeed, translating a quantum algorithm from one formalism to the other requires only polynomial efforts (for instance, in the number of additionally required qubits or quantum gates) and is therefore possible in principle. Nevertheless, from the programmer's point of view, either approach requires its own distinct form of thinking and of formalizing problems. We next elaborate on the underlying principles.

3.1 Adiabatic Quantum Computing

Adiabatic quantum computing (AQC), often also called *adiabatic quantum optimization* (AQO), relies on yet another quantum mechanical phenomenon. This phenomenon is formalized in the *adiabatic theorem* [65] which states that, if a quantum system starts in the *ground state* of a *Hamiltonian operator* which then gradually changes over a period of time, the system will end up in the ground state of the resulting Hamiltonian. Since Hamiltonians are energy operators, their ground states correspond to the lowest energy state of the quantum system under consideration. The adiabatic theorem therefore characterizes a form of *quantum tunneling* which refers to the fact that quantum systems can tunnel through energy barriers.

To harness AQC for problem solving, one prepares a system of qubits in the ground state of a simple, problem independent Hamiltonian and then adiabatically evolves it towards a Hamiltonian whose ground state corresponds to a solution to the problem at hand [66]. This can be done on D-Wave computers [67, 68, 69] which are particularly tailored towards solving *quadratic unconstrained binary optimization* problems (QUBOs) of the form

$$\boldsymbol{s}^* = \underset{\boldsymbol{s} \in \{-1,+1\}^n}{\operatorname{argmin}} \; \boldsymbol{s}^\intercal \boldsymbol{Q} \, \boldsymbol{s} + \boldsymbol{q}^\intercal \boldsymbol{s}. \tag{18}$$

Here, the 2^n different bipolar vectors $\boldsymbol{s}$ over which to minimize represent possible global states of a system of n entities each of which might be in one of two local states ($+1$ or -1). The coupling matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$ models internal interactions and the vector $\boldsymbol{q} \in \mathbb{R}^n$ models additional constraints or external influences.

In physics, QUBOs are known as Ising energy minimization problems [70] and, in machine learning, they are fundamental to the theory of Hopfield networks [71]. QUBOs pose discrete- or combinatorial optimization problems which are NP-hard and therefore notoriously difficult to solve. They also are surprisingly universal and of considerable practical importance since they often arise in the context of subset selection- or bipartition problems.

Given a set $\mathcal{X}$ of n elements x_i, *subset selection problems* ask for a subset $\mathcal{X}' \subset \mathcal{X}$ such that the elements of $\mathcal{X}'$ meet certain criteria and *bipartition problems* ask for a partition $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ such that the elements of either of the disjoint subsets $\mathcal{X}_1$ and $\mathcal{X}_2$ meet certain criteria. Now, if a subset selection- or bipartition problem can be (re)written as a QUBO over bipolar vectors $\boldsymbol{s}$, the entries of the solution $\boldsymbol{s}^*$ can be understood as indicator variables. For a subset selection problem, they define the sought after subset as $\mathcal{X}' = \{x_i \in \mathcal{X} \mid s_i^* = +1\}$ and, for a bipartition problem, we would have $\mathcal{X}_1 = \{x_i \in \mathcal{X} \mid s_i^* = +1\}$ and $\mathcal{X}_2 = \{x_i \in \mathcal{X} \mid s_i^* = -1\}$, respectively.

Subset selection- or bipartition problems of this kind abound in data mining, pattern recognition, machine learning, or artificial intelligence. General use cases can be found in data base search, the computation of medoids in k-medoids clustering, or the identification of support vectors in support vector machine training. Prominent practical applications where QUBOs are central include verification-, planning-, or assignment problems in areas such as logistics or finance [72, 73, 74, 75].

What renders quantum computing an attractive approach for dealing with difficult QUBOs is that seminal work by Farhi et al. [76] provided a simple general recipe for how to solve them via adiabatic evolution.

Assuming the parameters $\boldsymbol{Q}$ and $\boldsymbol{q}$ of a QUBO to be given, the basic idea is to consider a time dependent system of n qubits

$$|\psi(t)\rangle = \sum_{i=0}^{2^n-1} \alpha_i(t)\, |\psi_i\rangle. \tag{19}$$

which evolves under a time-dependent Hamiltonian $H(t)$ so that its behavior is governed by the Schrödinger equation

$$i\hbar \frac{d}{dt}\, |\psi(t)\rangle = H(t)\, |\psi(t)\rangle \tag{20}$$

The qubits are prepared in the ground state of a *beginning Hamiltonian* H_B which gradually changes towards the *problem Hamiltonian* H_P. For the latter, [76] proposes

$$H_P = \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{ij}\, \sigma_z^i\, \sigma_z^j + \sum_{i=1}^{n} q_i\, \sigma_z^i \tag{21}$$

where $\sigma_z^i = I \otimes \ldots \otimes I \otimes \sigma_z \otimes I \otimes \ldots \otimes I$ denotes the Pauli spin matrix σ_z acting on the i-th qubit. For the beginning Hamiltonian, [76] suggests

$$H_B = -\sum_{i=1}^{n} \sigma_x^i \tag{22}$$

where σ_x^i is the Pauli spin matrix σ_x acting on the i-th qubit. Considering an evolution from $t = 0$ to $t = T$, the Hamiltonian in (20) is assumed to be a convex combination

$$H(t) = \left(1 - \tfrac{t}{T}\right) \cdot H_B + \tfrac{t}{T} \cdot H_P \tag{23}$$

and can be used to evolve $|\psi(t)\rangle$ from $|\psi(0)\rangle$ to $|\psi(T)\rangle$ where $|\psi(0)\rangle$ is the ground state of H_B.

Finally, at time T, a measurement is performed on the qubits. This causes the whole system to collapse to one of its 2^n basis states and the probability for this state to be $|\psi_i\rangle$ is given by the amplitude $|\alpha_i(T)|^2$. However, since the adiabatic evolution was steered towards the problem Hamiltonian H_P, basis states $|\psi_i\rangle$ that correspond to ground states of H_P are more likely to be found.

The efficiency of this computational paradigm depends on the choice of T which is known to depend on the minimum energy gap between the ground- and first excited state of $H(t)$. While exponentially small gaps will render adiabatic quantum computing inefficient, problems with gaps that scale inverse polynomially can be solved efficiently. In fact, it is known that energy gaps are inversely proportional to the square root of the number of basis states that have energies close to global minimum [77]. For problems with a small number of valid solutions, an appropriate choice is $T \in O(\sqrt{2^n})$ [78]. This, in turn, is to say that adiabatic quantum computing can solve certain binary optimization problems quadratically faster than classically possible.

To illustrate the steps involved in setting up practical problems for solution on an adiabatic quantum computer and to better explain the process of adiabatic optimization, we briefly discuss an application example.

The *max-sum diversification problem* is of importance in location- or portfolio optimization and also arises in the context of information retrieval, recommendation, clustering, and other settings where one has to identify very distinct elements in a given data set. To be more specific, given a set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and an appropriate, problem specific distance measure $d(\cdot,\cdot)$, max-sum diversification asks for a subset $\mathcal{S}^* \subset \mathcal{X}$ of $k < n$ elements of maximum dispersion and thus consists in solving

$$\begin{aligned} \mathcal{S}^* = \underset{\mathcal{S} \subset \mathcal{X}}{\text{argmax}} & \sum_{\boldsymbol{x}_i \in \mathcal{S}} \sum_{\boldsymbol{x}_j \in \mathcal{S}} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \quad & |\mathcal{S}| = k. \end{aligned} \tag{24}$$

Collecting the $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in a distance matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ and considering a binary indicator vector $\boldsymbol{z} \in \{0,1\}^n$ where $z_i = 1$ if $\boldsymbol{x}_i \in \mathcal{S}$ and $z_i = 0$ otherwise and letting $\boldsymbol{1} \in \mathbb{R}^n$ denote the vector of all ones, the problem in (24) can also be written as

$$\begin{aligned} \boldsymbol{z}^* = \underset{\boldsymbol{z} \in \{0,1\}^n}{\text{argmax}} \; & \boldsymbol{z}^\intercal \boldsymbol{D} \, \boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{1}^\intercal \boldsymbol{z} = k \end{aligned} \tag{25}$$

Furthermore, when introducing a generic Lagrange multiplier λ and using the fact that $\boldsymbol{z}^\intercal \boldsymbol{D} \, \boldsymbol{z}$ is quadratic in $\boldsymbol{z}$, this NP-complete integer programming problem can be written as a QUBO over $\boldsymbol{z}$, namely

$$\boldsymbol{z}^* = \underset{\boldsymbol{z} \in \{0,1\}^n}{\text{argmin}} \; -\boldsymbol{z}^\intercal \boldsymbol{D} \, \boldsymbol{z} + \lambda \left(\boldsymbol{1}^\intercal \boldsymbol{z} - k\right)^2 \equiv \underset{\boldsymbol{z} \in \{0,1\}^n}{\text{argmin}} \; -\boldsymbol{z}^\intercal \boldsymbol{P} \, \boldsymbol{z} - \boldsymbol{p}^\intercal \boldsymbol{z} \tag{26}$$

However, the decision variables (18) are bipolar vectors $\boldsymbol{s} \in \{-1,+1\}^n$ whereas (26) minimizes over binary vectors $\boldsymbol{z} \in \{0,1\}^n$. We therefore emphasize a fact often used when (re)writing problems for adiabatic quantum computing, namely that it is easy to convert between binary and bipolar vectors, because $\boldsymbol{z} = (\boldsymbol{s} + \boldsymbol{1})/2 \Leftrightarrow \boldsymbol{s} = 2 \cdot \boldsymbol{z} - \boldsymbol{1}$.

Hence, when plugging $\boldsymbol{z} = (\boldsymbol{s} + \mathbf{1})/2$ into (26), it becomes an exercise in algebra [79] to show that max-sum diversification consists in solving the following energy minimization problem

$$\boldsymbol{s}^* = \underset{\boldsymbol{s} \in \{-1,+1\}^n}{\operatorname{argmin}} \; -\boldsymbol{s}^\intercal \boldsymbol{Q}\, \boldsymbol{s} + \boldsymbol{q}^\intercal \boldsymbol{s} \tag{27}$$

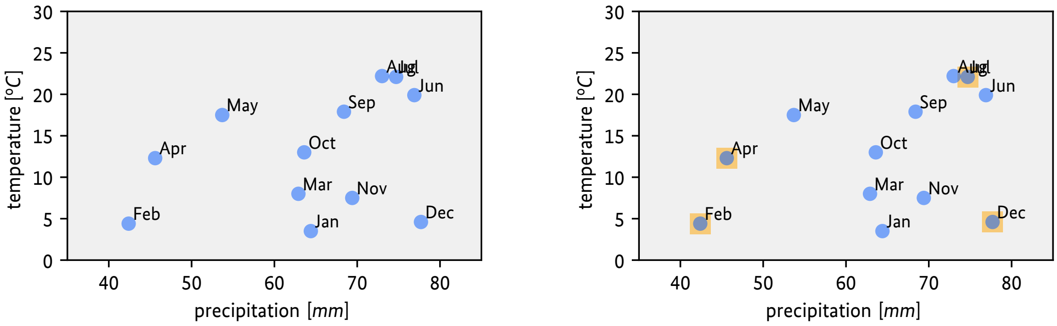
whose parameters amount to $\boldsymbol{Q} = -\frac{1}{4}\left(\boldsymbol{D} - \lambda\, \mathbf{1}\mathbf{1}^\intercal\right)$ and $\boldsymbol{q} = -\frac{1}{2}\left(\boldsymbol{D} - \lambda\, \mathbf{1}\mathbf{1}^\intercal\right)\mathbf{1} - \lambda\, k\, \mathbf{1}$. With respect to the original formulation in (24), we note that, once the solution $\boldsymbol{s}^*$ to (27) has been found, entries $s_i^* = +1$ indicate which $\boldsymbol{x}_i \in \mathcal{X}$ to select into $\mathcal{S}^*$. Figure 1 shows a specific, rather didactic instance of the problem of max-sum diversification and illustrates the inner workings of the AQC approach towards solving it. The data in Fig. 1(a) consists of a set $\mathcal{X}$ of $n = 12$ data points $\boldsymbol{x}_i \in \mathbb{R}^2$, each of which represents monthly climate conditions (average temperature and precipitation) in the city of Hamburg. Given this data, the task is to determine $k = 4$ months of diverse climatic characteristics.

To set up the QUBO parameters $\boldsymbol{Q}$ and $\boldsymbol{q}$, the data was normalized to zero mean and unit variance (a common step in machine learning). After this normalization, Euclidean distances $D_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ between far apart data points are of the order of $\mathcal{O}(2)$ so that setting $\lambda = 2\,n$ will cause neither of the terms in (27) to dominate the minimization problem.

Given $\boldsymbol{Q}$ and $\boldsymbol{q}$, one can prepare the beginning- and problem Hamiltonian according to the above recipe and then let an $n = 12$ qubit system $|\psi(t)\rangle$ adiabatically evolve over $T \in \mathcal{O}(\sqrt{2^{12}})$ time steps. To visualize the dynamics of this adiabatic quantum search for a solution, the process was simulated on a digital computer and the quantum computing toolbox QuTiP [80] was used to solve the Schrödinger equation in (20).

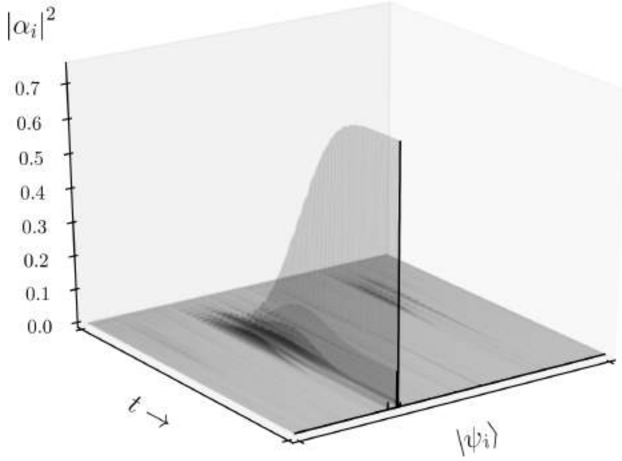
Figure 1(c) shows how the qubit system evolves after it has been prepared in a superposition of $2^{12} = 4096$ basis states $|000000000000\rangle, |000000000001\rangle, \ldots, |111111111111\rangle$ each of which represents a subset of the given set of data points. In particular, the figure visualizes the behavior of the amplitudes $|\alpha_i(t)|^2$ of the states the system can be measured in. At time $t = 0$, all basis states are equally likely but over time their amplitudes begin to diverge; amplitudes of basis states that correspond to low energy states of the problem Hamiltonian increase while amplitudes of basis states that could hardly be considered a solution to the problem decrease. At $t = T$, certain basis states are therefore more likely to be measured than others and the table in Fig. 1(d) ranks the ten most likely ones.

The most likely final state in this example is $|010100100001\rangle$ which, when understood as an indicator vector, indexes the subset consisting of the months of February, April, July, and December which are indeed diverse with respect to their climatic conditions (see Fig. 1(b)). Interestingly, the next most likely solution $|010100010001\rangle$ would swap July for August and, looking at the data in Fig. 1, this behavior of the quantum computing algorithm appears reasonable.



(a)

(b)



$\lvert\psi_i\rangle$	$\lvert\alpha_i\rvert^2$
$\lvert 010100100001\rangle$	0.742487
$\lvert 010100010001\rangle$	0.104235
$\lvert 010010100001\rangle$	0.020218
$\lvert 010101000001\rangle$	0.010994
$\lvert 010010010001\rangle$	0.002660
$\lvert 100101010001\rangle$	0.002426
$\lvert 100000001110\rangle$	0.002065
$\lvert 100101100001\rangle$	0.001764
$\lvert 011011000001\rangle$	0.001629
$\lvert 010011000101\rangle$	0.001400

(c)

(d)

Figure 1: Adiabatic quantum computation for max-sum diversification. (a) A didactic data set of $n = 12$ two-dimensional data points representing monthly climate conditions in the city of Hamburg. (b) Result obtained for $k = 4$ and Euclidean distances between data points. (c) Visualization of the evolution of the amplitudes of a corresponding 12 qubit system $|\psi(t)\rangle$. During its evolution over time t, the system is in a superposition of $2^{12} = 4096$ basis states $|\psi_i\rangle$ which represent potential solutions to the problem. Initially, each potential solution (reasonable or not) is equally likely. At the end of the process, one basis state has a noticeably higher amplitude $|\alpha_i|^2$ than the others and is thus more likely to be measured. (d) Ranking of the ten most likely states for the qubit system to be found in at the end of the adiabatic evolution; the likeliest final state indexes the months of February, April, July, and December and thus represents the solution shown in (b).

3.2 Quantum Gate Computing

Beside AQC, *Quantum Gate Computing* (QGC) is the dominant paradigm for quantum information processing systems. A detailed introduction into that topic can be found in [81].

Whether we are using qubits or bits, we need to manipulate them in order to turn the inputs we have into the outputs we need. Thus, QGC borrows many of its defining properties from classical computers: On a low level, digital computers process information by manipulating sets of bits via atomic logical operations between them. Input bits, being either in the state 0 or 1, are processed by so-called gates. Each gate performs a fundamental logical operation, e.g., `AND`, `OR`, `XOR`, `NOT`. For computations on a few bits, it is useful to represent this process in a diagram known as a circuit diagram or just circuit. These have input bits on the left, output bits on the right, and gates represented by specific symbols in between. An exemplary circuit that computes the difference of two bits, given a carry bit, is shown in Fig. 2 (a).

The evolution of a closed quantum system is described by a unitary transformation. Thus, all operations that we perform on qubits must be unitary too. In theory and practice, such operators typically act on only one or two qubits at a time but can be sequenced to form more complicated qubit transformations. Borrowing terminology from digital computing, quantum operators acting on qubits are also called *quantum gates*. An exemplary quantum gate circuit that creates a Bell state is shown in Fig. 2 (b).

A crucial difference to the classical setting is that, being unitary operators, quantum gates have to have the same number of input and outputs. Moreover, since the effect of any unitary gate U can be reversed by its Hermitian conjugate $U^\dagger$, a quantum gate's input can always be reconstructed from its output. For many classical gates such as the `AND` gate, this is not the case. At first sight, this reversibility constraint on quantum gates therefore seems to render them less expressive or powerful than classical ones. However, by introducing (several) ancillary bits, irreversible classical gates can be made reversible and there exist universal sets of reversible primitives [82]. Reversibility does therefore not restrict computability. Quantum gates can instead be seen as a generalization of classical reversible gates and the use of ancillary qubits is common practice in quantum computing.

Let us quickly recall the frequently appearing quantum gates and their relation to classical digital logic. The X-gate constitutes the counterpart of the logical `NOT`-gate. Applying

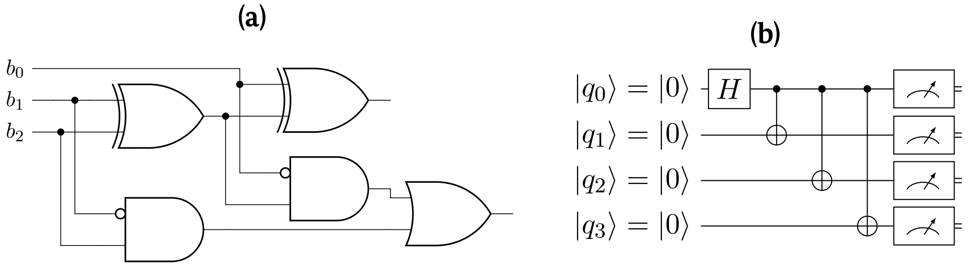

(a)
b_0
b_1
b_2
(b)
$|q_0\rangle = |0\rangle$
$|q_1\rangle = |0\rangle$
$|q_2\rangle = |0\rangle$
$|q_3\rangle = |0\rangle$
H

Figure 2: Left: Example of a digital subtractor circuit. Right: Example of a quantum gate circuit for generating a 4-qubit Bell state.

X to a qubit switches the amplitudes of $|0\rangle$ and $|1\rangle$.

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

Closely related are the Y-gate

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0|$$

and the Z-gate

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \ .$$

At a first glace, X, Y, and Z have not much in common. However, each of them can be interpreted as a rotation around the corresponding axis (x-axis, y-axis, z-axis) by π radians in a conceived 3-dimensional coordinate system. However, applying the Z-gate to one of the basis states $|0\rangle, |1\rangle$ seems to have no effect:

$$Z|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \qquad Z|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = -|1\rangle$$

Clearly, $|0\rangle$ and $|1\rangle$ are eigenstates of Z. They form the so-called Z-basis. There is, in fact, an infinite number of bases from which some have specific names in the context of quantum gate computing. One that appears directly in various quantum algorithms is the X-basis, constituted by $|+\rangle$ and $|-\rangle$.

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The specific reason for why this basis appears frequently is the Hadamard gate, also known as H-gate. It allows us to create a superposition of $|0\rangle$ and $|1\rangle$, effectively realizing a uniform distribution over $\{0, 1\}$.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Interpreted as a change of basis, it allows us to move from the Z-basis to the X-basis:

$$H|0\rangle = |+\rangle \qquad H|1\rangle = |-\rangle$$

An important insight for digital circuits is the universality of specific gate sets. There is no need to have hardware implementations of all possible logical 2-bit gates available—it can be shown that a single `NAND`-gate suffices to implement all Boolean operations. Similar relations can be found for quantum gates as well, e.g. the following equivalence:

$$HZH = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = X$$

Indeed, there is a finite number of possible logic gates. However, quantum circuits allow for *parametrized gates*. These gates require numeric parameters to define their actual function. They are of utmost importance for quantum machine learning in general, since they allow us to define families of functions via a single circuit. Such families constitute model classes for which classical machine learning techniques can be applied to select those models which fit best to some user specified data. We will revisit this topic in Section 5.

The P-Gate (also known as *phase gate*) generalizes the Z-gate. It performs a rotation by ϕ radians around the Z-axis.

$$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\phi) \end{bmatrix}$$

Due to closedness, all quantum gates can in fact be absorbed into a single canonical representation. This U-gate is parameterized and subsumes all possible single qubit gates.

$$U(\theta, \phi, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\exp(i\lambda)\sin\left(\frac{\theta}{2}\right) \\ \exp(i\phi)\sin\left(\frac{\theta}{2}\right) & \exp(i(\phi+\lambda))\cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

All gates discussed so far act only on a single qubit. However, multi-qubit operations are required to realize non-trivial algorithms and to facilitate quantum entanglement between qubits. The most common way to work with multiple qubits is via *controlled not* (CNOT) gates. They are the quantum counterpart to the classical XOR-gate. While single qubit gates can be fully described by a single matrix, CNOT has two representations,

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

depending on which qubit is the controller and which is the target. The semantic of the CNOT is as follows: if the control-qubit is in state $|0\rangle$, the target remains unchanged. If the control-qubit is in state $|1\rangle$, an X-gate (NOT-gate) is applied to the target. This effect can be applied to create a Bell state over n qubits: One initializes all qubits to $|0\rangle$ and applies a H-gate to one of them, say, q_0. q_0 is then in a uniform superposition of $|0\rangle$ and $|1\rangle$. Then, $n-1$ CNOT-gates are added, connecting q_0 with each of the remaining $n-1$ qubits. In all cases, q_0 is the controller. Now, measuring any of the n qubits in state $|0\rangle$ (or $|1\rangle$) implies that q_0 must be $|0\rangle$ (or $|1\rangle$) too, and hence all other qubits. In other words, measuring any of the n qubits determines the state of all remaining qubits. The resulting circuit is depicted in Fig. 2 (b) for $n = 4$.

As mentioned earlier, the action of any quantum circuit can be written as a product of unitaries.

$$U = U_1 U_2 U_3 \dots U_d$$

Here, d is called the *depth* of the circuit. Formally, each 1-qubit and 2-qubit operator has to be extended to an 2^n-dimensional operator in order to fit into the above expression.

Measurements are special in that they enforce the quantum state to collapse into a classical one. The state will lose its probabilistic nature and all subsequent measurements of the outputs will yield the same result deterministically. A simple but important example of a measurement is that of a qubit in the computational basis. This is a measurement on a single qubit with two outcomes defined by the two measurement operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. We observe that each measurement operator is Hermitian, and that $M_0^2 = M_0$ as well as $M_1^2 = M_1$. Thus, $I = M_0^\dagger M_0 + M_1^\dagger M_1 = M_0 + M_1$. Suppose the state being measured is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Then, the probability of obtaining the measurement outcome 0 is

$$\mathbb{P}(0) = |\psi\rangle M_0^\dagger M_0 \langle\psi| = |\psi\rangle M_0 \langle\psi| = |\alpha|^2$$

Similarly, the probability of obtaining the measurement outcome 1 is $\mathbb{P}(1) = |\beta|^2$.

3.3 Further Quantum Computing Paradigms

The above discussion might lead to a misconception, namely that adiabatic quantum computing and quantum gate computing are the only “paradigms” in the broad field of quantum information processing. This short section is therefore supposed to help avoid possible confusion. The following briefly discusses the notions of topological quantum computing and variational quantum computing. The latter is of considerable interest in the context of quantum machine learning and will be discussed in detail in section 5.

3.3.1 Topological Quantum Computing

Another term ostensibly similar to adiabatic quantum computing or quantum gate computing is *topological quantum computing* [83]. However, whereas the former two refer to logical aspects of quantum computing and abstract away from how to physically implement logical qubits, the latter refers to physical aspects and a physical implementation of qubits.

We therefore only briefly mention that a topological quantum computer is still a mainly theoretical device that considers two-dimensional quasi-particles called anyons, their braids, and their characteristics in three-dimensional space-time. If such a quantum computer could be built, for instance utilizing quantum Hall effects, an expected advantage over currently common technical realizations based on, say, superconducting circuits or trapped ions, is that it would be more robust against decoherence.

3.3.2 Variational Quantum Computing

Variational quantum algorithms (VQA) and variational *quantum approximate optimization algorithms* (QAOA) combine quantum computations with classical computations in iterative feedback loops [84, 85]. This is supposed to address limitation of current NISQ devices (limited numbers of qubits, limited circuit depths, measurement noise) and aims at harnessing the best of both worlds (classic and quantum).

Either classical optimizers are used to design working quantum circuits, or quantum computation outcomes are measured and classical analysis is used to update Hamiltonian operators for the next round of quantum computations. By now, VQAs have been proposed for many applications and they appear to be the currently best way of exploiting quantum advantages in the NISQ era [86]. Indeed, one can show that variational quantum computing is universal and that efficient input/output strategies for looped classic-to-quantum optimization are possible [87]. Corresponding methods are also known as hybrid quantum-classical computing methods and we will return to these ideas in section 5.

3.4 Technical State of the Art and Limitations

As of this writing, most technologies used to physically implement qubits still face issues of stability, decoherence, error tolerance, and scalability. As a consequence, current NISQ devices therefore still need many physical qubits just for the purpose of error-correction. In other words, present day logical qubits typically consist of many physical qubits in order to be able to perform useful computations. This can cause a gap between theory and practice of quantum algorithm design. For instance, theoretically valid algorithms which assume a large number of logical qubits to be available may not yet be practical.

Moreover, as of today, quantum computing is still bit level computing. That is, abstract data structures (such as linked lists or binary trees) or control structures (such as `if-then-else`-statements or `for-` or `while`-loops known from higher level programming languages or are not yet available to quantum programmers. Even common programming patterns such as variable assignments (`x = y` or `x = x+1`) are not possible on present day quantum computers because the *no-cloning theorem* and no-broadcast theorems state that it is impossible to create independent identical copies of arbitrary quantum states [88].

Although there are efforts towards identifying quantum programming patterns [89] and although (open source) software development kits for working with quantum computers such as Qiskit [90], Cirq [91], Forest [92] or PennyLane resulting from a collaboration of several smaller quantum computing companies [93] become increasingly available, it is important to note that these are tools for very low-level programming. For instance, the software development kits we just mentioned are Python libraries for the mathematical design of quantum circuits. They do allow for running quantum circuits on quantum computers or simulators but do not yet provide features equivalent to high-level data structures or control flow structures.

We will return to these issues and their impact on algorithm design for quantum machine learning in section 6.

4 Machine Learning in a Nutshell

Machine learning is the science of fitting parameterized mathematical models to data in order to realize intelligent systems that can make predictions or perform inference.

While the term *data* is often understood to mean known facts that can be recorded and have an implicit meaning, machine learning generally assumes a more technical point of view. In what follows, we, too, will assume this point of view and understand *data* to refer to collections of numeric values that can be obtained from measurements, surveys, interactions with technical devices, or comparable procedures and can be processed by computers. At first sight, this focus on numeric values may appear to restrict generality. Note, however, that everything that is held in the memory of a computer (a text, a piece of music, an image, ...) is encoded in terms of bit patterns and therefore in terms of numbers.

We will further assume that any individual data object can be encoded in terms of a tuple of m real numbers. Using a general notation, we will write such a tuple as $\boldsymbol{x} \in \mathbb{R}^m$ and call it a *data point*; a collection $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ of n data points will be called a *data set*.

Data can be annotated by *metadata* which contain additional information. Common everyday examples are: a time stamp indicating when a picture was taken, a caption describing the content of a figure, the resource description framework (RDF) tags of a Website, or object identifiers such as the international standard book number (ISBN) of a book. Again resorting to an abstract notation, we will express metadata as $\boldsymbol{y} \in \mathbb{R}^l$ and refer to an annotated data set $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1) \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ as a *labeled data set*.

In machine learning, data sets are seen as a source of information about a specific scenario. The fundamental assumption is that data which were gathered in a certain context or with respect to a certain application domain are not arbitrary. On the contrary, machine learners always suppose that application specific data sets exhibit certain regularities which, in general thought, may be intricate or complex and therefore not immediately obvious to human analysts.

In other words, any machine learning solution for a real world problem implicitly posits that there exists some generally unknown process or mechanism which can explain variations in appearance within a given sample of example data. Mathematically, this translates to the assumption that there exists some function $f \in \mathcal{F}$ with parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ which models salient as well as latent characteristics of a set of data. For instance, in the case of labeled data, this means

$$\boldsymbol{y}_j = f(\boldsymbol{x}_j \mid \boldsymbol{\theta}) + \varepsilon_j \tag{28}$$

where the *noise term* ε_j accounts for possible inaccuracies which may, for instance, be due to corrupted measurements or faulty annotations. The family of functions $\mathcal{F}$ to which the model f belongs to is called the hypothesis- or *model class* and $\boldsymbol{\Theta}$ denotes the corresponding *parameter space*.

Other categorical terms which commonly occur in discussion about machine learning are *supervised learning*, *unsupervised learning*, and *reinforcement learning*. These refer to general machine learning philosophies which differ with respect to the richness of data available for training and testing.

Supervised learning deals with labeled data $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ and the goal is to train a model such that it can produce an appropriate output $\boldsymbol{y}$ for a given input $\boldsymbol{x}$. A variant of this setting is *semi-supervised learning* which aims at assigning a label to unlabeled data using the knowledge contained in a small set of labeled data. If a semi-supervised learning system also incorporates its own earlier predictions into this assignment processes, it is sometimes referred to as a *corrective learning* system.

Unsupervised learning, on the other hand, works with unlabeled data $\boldsymbol{x}_j$ and aims at uncovering latent or inherent structures within a given data set. Examples of structures users might be interested in are clusters, higher order correlations, or relational dependencies.

Finally, reinforcement learning is a type of supervised learning that receives feedback in place of a label [94]. Based on such feedback, reinforcement learning methods tune models that are typically intended for decision making in feedback situations where a current output may impact the next input. Such a setting can often be formalized in terms of a partially observable Markov decision process (POMDP) [95]. Roughly speaking, learning in such settings happens in a (guided) trial and error procedure where the software agent is in some state, decides for an action which leads to a successor state, and only later finds out if a sequence of actions had a desired effect. This delayed feedback is then used to adjust the action selection mechanism via dynamic programming approaches and, over time, the agent learns which action to perform when in order to achieve certain (long term) goals.

In what follows, we give a more detailed account of the stages involved in training a machine learning system. We will largely focus on the case of unsupervised learning, yet, everything we discuss also applies to the other learning paradigms.

4.1 The Machine Learning Pipeline

Given the above premises, the process of developing a machine learning solution for a practical application can be broken down into several distinct phases. It begins with a data collection campaign in which representative examples for the intended application scenario are gathered and possibly annotated.

Here, it is of utmost importance to insist on representative data. This means the collected data points have to cover every foreseeable situation that may occur during the later deployment of the system. Indeed, a common and easily avoidable cause for insufficient reliability of machine learning solutions are biased training data which omit or neglect some of the inputs a system is expected to handle. In practice, this can have embarrassing consequences (for instance in incidents where software classified pictures of African Americans to depict gorillas[1]) to downright catastrophic ramifications (such as when a cars autopilot software failed to recognize a truck, crashed into it, and caused the death of its driver[2]).

The crucial point is that if a deployed system is confronted with a kind of input which has been overlooked during its development, its output can be unpredictable. Ideally, the collected data would be balanced and reflect all possible use case situations in about equal parts. This, however, may not always be possible. For example, in a predictive maintenance scenario where the task is to evaluate sensor data in order to predict whether a machine will continue to run smoothly or whether a failure is imminent, there may be much fewer examples of failure cases than of normal conditions.

Next, there may be a data pre-processing phase in which the collected data might be brought into a form more amenable for processing or where flawed data points might be filtered out. For instance, text data might be transformed into an appropriate numeric representation or measurement noise in sensor data might be smoothed.

After pre-processing, the collected data is split into two disjoint subsets, namely a set of *training data* and a set of *test data* and developers have to decide for a model class. This decision usually requires some domain expertise and will generally depend on the nature of the data as well as the intended use case. For instance, when dealing with a time series prediction scenario, one might opt for a polynomial function, a Markov chain, a Gaussian process, a decision tree, or a recurrent neural network. Each such general decision is followed by more specific decisions: Which degree is the polynomial supposed to have? How many states should the Markov chain contain? How to parameterize the kernel of the Gaussian process? How deep should the decision tree be? How many neurons should the neural network have and how should they be interconnected?

As these examples indicate, there often are numerous mathematical models that may apply to a given problem. Alas, clear cut criteria or simple suggestions for which model to use when are hard to come by. Rather, substantial research efforts are spent on understanding the usefulness of different models in different contexts and on developing new models for new contexts. However, a general recent trend is to avoid overly specialized models but to consider very flexible and thus widely applicable models such as deep neural networks instead.

In the *training phase*, the parameters of the chosen model are then adjusted such that the model matches the training data to the best extend possible. This adjustment happens automatically by means of running *learning algorithms* which are typically based on (sta-

[1] see, for example, https://www.forbes.com/sites/mzhang/2015/07/01/google-photos-tags-two-african-americans-as-gorillas-through-facial-recognition-software/ or https://www.nytimes.com/2021/09/03/technology/facebook-ai-race-primates.html

[2] see, for example, https://arstechnica.com/cars/2019/05/feds-autopilot-was-active-during-deadly-march-tesla-crash/ or https://www.forbes.com/sites/bradtempleton/2020/06/02/tesla-in-taiwan-crashes-directly-into-overturned-truck-ignores-pedestrian-with-autopilot-on/

tistical) optimization techniques. In order for this kind of mathematical learning to be possible at all, there has to be a criterion for how to measure how well the model and its current choice of parameters matches the training data. In other words, there has to be an objective for the parameter optimization process and these objectives are often formalized in terms of an error- or *loss function*. A simple example of such a loss function often used when learning from labeled data is the mean squared error

$$E(\boldsymbol{\theta}) = \frac{1}{n} \sum_{j=1}^{n} \| \boldsymbol{y}_j - f(\boldsymbol{x}_j \mid \boldsymbol{\theta}) \|^2 \tag{29}$$

and the corresponding training objective would be to solve

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\text{argmin}} \; E(\boldsymbol{\theta}) \tag{30}$$

Note, however, that potential loss functions are as numerous as potential model choices and that loss functions should, in fact, be chosen with respect to the data and model at hand. For example, when working with binary labels, one often considers the so called hinge loss or, when training probabilistic models, one might want to maximize a likelihood. Even other loss functions are based on divergence measures, entropy criteria, or problem- or model specific distances. In fact, the investigation of loss functions, their characteristics, and applicability is an important topic of ongoing machine learning research.

Learning algorithms are numerous, too, and the question of which algorithm to consider in the training phase should again be decided with respect to the given data, model, and loss function. Indeed, depending on the chosen model and loss function, it might be trivial to neigh impossible to optimally solve the learning problem in (30). For instance, if $f(\boldsymbol{x} \mid \boldsymbol{\theta})$ is linear in its parameters $\boldsymbol{\theta}$ and $E(\boldsymbol{\theta})$ is convex, there will be a closed form solution $\boldsymbol{\theta}^*$. Depending on the numbers of training data points and model parameters, it may still require considerable efforts to practically compute this solution, but one knows that it exists. On the other hand, if the model $f(\boldsymbol{x} \mid \boldsymbol{\theta})$ is highly non-linear, the error landscape $E(\boldsymbol{\theta})$ will have numerous local minima and an algorithm that is guaranteed to find the globally optimal solution $\boldsymbol{\theta}^*$ might not even exist. In situations like these, one often applies optimization techniques such as gradient descent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \cdot \nabla E(\boldsymbol{\theta}_t) \tag{31}$$

or other iterative methods. Note, however, that $f(\boldsymbol{x} \mid \boldsymbol{\theta})$ may be a complicated, composite function so that the computation of $\nabla E(\boldsymbol{\theta}_t)$ itself may already require a whole chain of complex operations (such as in case of the *backpropagation* algorithm for training neural networks). Moreover, the choice of meta-parameters of a training algorithm such as the step size η_t in (31) is typically a non trivial matter and might require careful tuning. All in all, it can thus be very burdensome to train expressive models with many degrees of freedom or parameters. In fact, modern models such as deep neural networks with billions of parameters, typically necessitate the use of high performance computing hardware for training. Improved training algorithms, their convergence rates, and performance guarantees are therefore yet another topic of ongoing machine learning research.

In the *test phase*, the trained model is then validated and evaluated on the test data set. This requires an appropriate performance measure for which there are again abundantly many (application dependent) choices. For instance, in the context of a classification task, one often evaluates the accuracy, i.e. the percentage of correct class predictions, of a trained classifier. For an information retrieval system, one may be interested in its recall and precision which measure the percentage of relevant retrieved instances and the percentage of relevant instances among the retrieved instances. But there also exist highly specialized performance indicators such as the perplexity of a language model or the BLEU score of a language translation system.

What is of pivotal importance in this phase is that training data and test data must be independent. In other words, data points considered during training must not be reconsidered during testing. This is crucial because what really needs to be assessed are the *generalization* capabilities of the trained model. That is, one needs to evaluate how well the function $f(\boldsymbol{x} \mid \boldsymbol{\theta}^*)$ performs on novel data, i.e. on data it has not seen during training. For reasons we will explain below, it is actually dangerous to evaluate the trained model on its training data and, for reasons we explained above, it is again pivotal that the testing data are representative.

Once a trained model has been thoroughly tested and found to perform reliably as well as accurately, it can finally be deployed in practice and be released into its *application phase*.

Use cases for this general methodology are manifold but most commonly found in settings where systems need to make data-based predictions, generate data-based suggestions or decisions, or classify novel observations or measurements. We also note that terms such as prediction, classification, or decision making can have a rather broad meaning. For instance, a prediction could be an estimate of tomorrow's stock market closing price or the statement that this English sentence translates into that German sentence. A classification could be as simple as "this picture shows a cat" or as sophisticated as "this picture shows a little boy on a sunny beach playing with a red ball". What kind of capabilities a learning system can achieve largely depends on the nature of the available training data and on the nature of the chosen model. With respect to the latter, notable strides have been made using modern neural network models and architectures such as variational auto-encoders, generative adversarial networks, or transformer networks [96, 97, 2].

4.2 Lazy vs. Eager Machine Learning

Yet another categorization of machine learning techniques contrasts *lazy learners* with *eager learners*. It is worthwhile to briefly mention this distinction as it points to different kinds of levers one may consider when trying to incorporate quantum computing techniques into the machine learning pipeline.

A lazy learner is a machine learning system that simply stores data and delays modeling until asked to make predictions. Such approaches are particularly suitable when dealing with large and frequently changing databases since only the affected part of the model needs to be retrained for new data. A simple yet common example of a lazy learner for classification problems is the k-nearest neighbor method which classifies novel observations based on their distances to known, i.e. previously learned, prototypes. Given an incoming data point, it searches a database of prototypes for the k nearest ones, performs a (weighted) majority voting over their class labels, and assigns the result to the new observation. Here, training is rather simple and can usually be accomplished quickly as the training problem basically consists in automatically identifying suitable prototypes. Application of such a classifier is usually simple and not too time consuming as well. However, depending on the size of the prototype database and the number k of nearest neighbors to consider, the required computations be burdensome and may necessitate the use of specific data- or index structures to guarantee fast runtimes. A considerable advantage of methods such as a k-nearest neighbor classifier is that they can easily be (re)trained during the application phase of the system because their prototypes may be extended or modified.

An eager learner, on the other hand, is an algorithm that trains a model in a dedicated offline training phase and considers a usually large but static training data set. This can lead to a very tedious or computationally intensive learning phase, but the application of the trained model to new examples is very efficient. Most of the well known machine learning models such as, say, neural networks are trained eagerly. Indeed, since we just mentioned prototype techniques as a prime example for lazy learning, it is important to note that not all prototype-based models are lazy. A prominent example of a class of eager prototype-based models are support vector machines.

4.3 Sources of Uncertainty in Machine Learning

To conclude this short overview of machine learning, we need to point out that the process of fitting a parameterized mathematical model to data is inherently statistical. This statement applies to any machine learning paradigm and should be understood as follows: From an abstract point of view, a fitted model provides a summary or compressed representation of the information contained in a training sample and there are several uncertainties involved in its training that may lead to different results in different training runs.

First of all, data collection is an informed but random process since different machine learning practitioners may collect different data samples when tackling the same problem. Second of all, modelling is an informed but random process since different practitioners may opt for different models when tackling the same problem. Third of all, model training may start with random initializations of the model parameters and thus settle in different local minima of the chosen loss function.

These sources of uncertainty can lead to undesirable outcomes. For instance, models can be biased and inappropriate for the data at hand. This is mainly the case when models are too simple or not flexible enough to capture relevant input output relations. In this case, training will lead to *underfitting*. Models can also be too flexible and thus overly sensitive to small fluctuations in their training data. Training such models on (slightly) different data samples will likely produce considerably different results and they are said to show high variance. High variance models tend to learn minute irrelevant details and may therefore suffer from *overfitting*.

Phenomena like these explain the need for training and testing on independent data sets because only if training and test data differ can under- or overfitting be identified.

The so called bias-variance dilemma poses a considerable challenge in machine learning as one typically strives for models that capture patterns in the training data and also generalize well to unseen data. Alas, it may not be possible to achieve both these goals simultaneously since bias and variance tend to be reciprocal. As a rule of thumb, bias can be reduced by focusing on local information as it is done in nearest neighbor models or in radial basis function models. Variance, on the other hand, can be reduced through averaging over multiple data points or larger regions in data space as it is done in most other common, typically more complex models.

A common fallacy in this context is to assume that model complexity (measured in terms of the number of model parameters) causes variance and overfitting. However, this need not be the case; instead, overfitting in complex models is mainly due to too much freedom in the choice of their parameters. If this freedom is restricted, for instance by constraining the value a parameter can assume, overfitting can often be avoided. Imposing restrictions on model parameters is known as *regularization* and there exists a host of methods for how to accomplish this.

Other methods for mitigating variance include data dimensionality reduction, feature selection, or increasing the size of training data sets. At the same time, adding features or increasing data dimensionality can decrease bias. Moreover, many models and algorithms come with specific parameters which allow for trading off bias and variance. For instance, choosing a higher value of k in a k-nearest neighbor model will increase its bias and decrease its variance whereas a lower value of k will decrease bias and increase variance.

Yet another way of addressing the bias-variance dilemma consist in using *ensemble learning* techniques such as boosting or bagging. While boosting algorithms combine many models of individually high bias into an ensemble of low bias, bagging methods combine individual models of high variance into an ensemble with low variance. In this context, it is interesting to observe that recent empirical results indicate that modern neural networks with very wide layers (which can be seen as ensembles of individually weak learners) do not seem to suffer from the reciprocal bias-variance characteristics of more traditional models [98, 99].

5 Quantum Machine Learning

Modern machine learning is undeniably successful but also a very resource intensive endeavor. By now, the field has reached a point where practical computational efforts for training state-of-the-art models can only be dealt with on high performance computing hardware. Since this trend is likely to continue, it is no surprise that a growing number of machine learning researchers are beginning to look at the potential benefits of quantum computing.

By now, the scientific literature on quantum machine learning is vast and taking stock of the state of the field is warranted. In the following survey, we will consider quantum inspired classical algorithms for classical data, genuine quantum algorithms for classical data, classical algorithms for quantum data, and quantum algorithms for quantum data.

5.1 Quantum Inspired Machine Learning

Quantum inspired models constitute a broad class of frameworks, methods, and algorithms for classical data processing on classical computers that involve quantum mechanical concepts, in particular, the mathematics of quantum mechanics and quantum information processing. Alas, the understanding of what kind of methods and techniques are inspired by quantum mechanical insights varies widely. Some authors adhere to a narrow interpretation and only consider methods or algorithms which clearly would not exist without quantum mechanics. Others assume a rather broad point of view and consider, say, general optimization techniques such as simulated- or mean field annealing to be quantum inspired just because they occur in- or may have originated from the study of certain quantum mechanical systems.

An example of an early contribution that is quantum inspired in the narrow sense can be found in a book by van Rijsbergen in which he develops a quantum theory of information retrieval [100]. Modern information retrieval deals with the problem of searching data sets of unstructured media objects (texts, images, videos, etc.) for items or content related to a user query. A well known instance of this setting is Web search and Web search engines have become the best known examples of information retrieval systems.

The information retrieval problem is often formalized in terms of linear algebraic vector space models. Here, n data objects are encoded in terms of m dimensional vectors which are then gathered in an $m \times n$ matrix. A decomposition of this matrix into factor matrices of lower rank can reveal latent structures in the data and, if queries are encoded in terms of m dimensional vectors, too, query matching simply becomes the evaluation of inner products.

Based on these ideas and borrowing inspiration from quantum logic, a venerable quantum theoretic approach towards reasoning [101], van Rijsbergen argues that information retrieval should ideally be formulated in terms of superpositions in (infinitely dimensional) Hilbert spaces. There, eigenvectors of quantum density operators would represent basic (latent) concepts to be searched for, and the corresponding eigenvalues would measure overlaps between concepts and queries. His ideas are convincing and compelling but also of limited practical value. Suitable quantum hardware to implement them on does not yet exist and digital emulations that would go beyond toy examples are impossible, because they would require exponentially large amounts of digital memory to store adequate Hilbert space representations of media objects.

Examples of a more generous interpretation of quantum inspired algorithms can be found in a recent contribution by Arrazola et al. [102] who, among others, are concerned with recommendation systems. Recommendation systems are a topic closely related to information retrieval, and play an important role in e-commerce. Here, too, popular models are based on linear algebraic formulations. If user-item preferences are represented in terms of a typically sparse $m \times n$ matrix, the problem of recommending items to users becomes a problem of predicting missing matrix entries. Just as in the case of vector space information retrieval, this prediction problem can be tackled using matrix decomposition methods.

While matrix factorization problems frequently occur in computational intelligence applications, their exact solutions are often hard to come by because the amount of floating point operations required for modern large scale matrices exceeds what is reasonably possible on digital computers. Therefore, there exists a vast amount of literature on randomized or probabilistic algorithms for fast, approximate numerical linear algebra [103, 104, 105, 106, 107, 108].

Indeed, Arrazola and his colleagues consider the FKV sampling algorithm [103], replace remaining exact procedures (for the computation of eigenvalues) by expected value estimations, and deem this a quantum inspired approach because similar estimators occur in quantum mechanics. However, probabilistic approaches to matrix factorization (with proper renormalization) have a venerable history, among others in machine learning [104, 105, 109], where they have hardly been thought of as inspired by quantum mechanics. In fact, if one were to subscribe to the broad interpretation of quantum inspired techniques, any statistical inference involving, say, Monte Carlo sampling, would count as such just because its mathematical apparatus was first developed by the quantum physics community.

To be frank, the use of the term *quantum inspired* often appears to be a marketing instrument for making certain approaches look more interesting than they turn out to be upon inspection. Consider, for instance, methods surveyed by Zhang [110] who benchmarks more than a hundred quantum inspired evolutionary optimization algorithms. Evolutionary algorithms take inspiration from biology because they work on populations of genotypes (i.e. encoded possible solutions to a problem) which express phenotypes (i.e. correspondingly decoded solutions). Problem solving or optimization happens in an iterative manner, where current auspicious genotypes are recombined or mutated into new ones whose phenotypes then represent new possible solutions. Since better solutions are used to replace worse ones, the population as a whole becomes better over time. This exploration of the solution space continues until the population contains enough members which express acceptable solutions. Traditionally, genotypes are often just bit string encodings of more complex objects, and recombination and mutation happen in a purely random fashion. Obvious improvements over this baseline include more expressive encodings and more informed or probabilistically guided updates. Granted, some of the methods surveyed in [110] apply genuine quantum concepts (such as encodings in terms of sets of two-dimensional complex vectors or qubits) but many others simply rely on probabilistic update mechanisms whose connection to quantum mechanics has to be considered loose at best.

In what follows, we will therefore attempt to focus on quantum inspired techniques that are relevant to machine learning and, at the same time, are recognizably rooted in the mathematics of quantum mechanics and quantum computing.

5.1.1 Tang's Quantum Inspired Algorithm for Recommendation Systems

A broad class of techniques which are genuinely quantum inspired consists of methods that "dequantify" quantum algorithms. A widely reported example of such a method is due to Tang [111] who, as a graduate student, famously discovered a classical analogue of a quantum recommendation system algorithm introduced by Kerenidis and Praksh [112].

The quantum algorithm in [112] repeatedly samples from a low-rank matrix approximation by means of running quantum phase estimation [113, 114, 115] and quantum projections (partial measurements). This allows for sampling matrix elements proportional to their magnitude and thus for sampling matrix elements most relevant to the recommendation task without having to access every element of the matrix. Kerenidis and Praksh thought this quantum algorithm to be exponentially faster than classically possible, because a classical implementation of the procedure seemed to necessitate iterations over all matrix elements. Crucially, this supposed quantum advantage hinges on the assumption that incoming classical data can efficiently be encoded in terms of quantum states. However, while the QML literature often assumes such state preparations to be given and posits the existence of quantum random access memories (QRAMs) in which these states are available, Kerenidis and Praksh actually provided a protocol and data structure for this purpose.

Tang's fundamental insight was that this protocol also allows for encoding classical data in a representation that satisfies norm constraints required in randomized linear algebra. There, it is known that norm-based sampling of, say, matrix columns minimizes variance among all unbiased estimators of factor matrices. Moreover, robust estimates can be obtained from samples smaller than the size of the matrix to be factored [103, 108]. Tang therefore swapped quantum state preparation for preparation of a classical sampling procedure and thus obtained a classical algorithm only polynomially slower than the original quantum method. In other words, Tang's work showed that, in this particular matrix completion setting, potential quantum speedup is not as substantial as it appeared to be at first sight. This led her to conclude that claims as to speedups of QML algorithms over classical ML algorithms should take into consideration any state preparation assumptions in a QML model and match them against sampling assumptions in a corresponding classical ML model.

Tang's work therefore provides new directions for classical algorithm research and helps better understanding for which kind of problems one can or cannot expect exponential quantum speedup. Indeed, in the wake of her discovery, efforts in "dequantifying" quantum algorithms have noticeably increased and combinations of her encoding scheme with randomized numerical linear algebra are becoming ever more popular. Examples include new methods for estimating pseudoinverses [116] based on quantum algorithms for the singular value decomposition [117], sub-linear algorithms for solving linear systems of equations [118] based on the HHL algorithm [119], and even faster algorithms for recommendation systems and regression problems [120].

5.1.2 Tensor Networks

Tensor networks are mathematical models originally developed for the study of many-body quantum systems in condensed matter physics [121, 122]. More recently, they were found to be helpful tools in quantum information processing and connections to machine learning models have been established [123, 124].

Tensor networks represent quantum states based on local entanglement structures. This is particularly useful whenever one is dealing with states that have a tensor product structure. Consider a system of n qubits whose description would usually require $\mathcal{O}(2^n)$ complex coefficients or, equivalently, a complex-valued tensor $\mathbf{\Gamma}$ with n indices

$$|\psi\rangle = \sum_{i_1, i_2, \ldots, i_n} \gamma_{i_1 i_2 \ldots i_n} |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle \tag{32}$$

However, similar to the matrix factorization techniques discussed above, such tensors can be modeled in terms of contractions of products of lower order tensors, for example

$$\gamma_{i_1 i_2 i_3 i_4} = \sum_{j,k,l} \alpha_{i_1 jk}\, \beta_{k i_2 l}\, \delta_{l i_3 i_4} \tag{33}$$

This allows for graphically describing the tensor as a network of interconnected tensors and, interestingly, when modeling physical systems, such networks tend to have common basic building blocks.

Just as the matrix factorization techniques sketched above, low rank tensor decompositions have a venerable history in machine learning and data science [125]. Among others, they allow for flexible recommendation, fast video analysis, or robust document understanding [126, 127, 128]. Since data scientists increasingly deal with multidimensional data of high volume and higher order latent structures, tensor decompositions and tensor networks together with corresponding inference- or learning algorithms are attracting more and more interest. There exist kernelized or non-Euclidean extensions, various cost- or loss functions to guide the estimation of meaningful factors, and distributed computing approaches. Tensor networks thus allow for addressing large scale problems and are at the heart of generalized regression and classification techniques, support tensor machines, higher order canonical correlation analysis, higher order partial least squares, and generalized eigenvalue decomposition. Last but not least, they also allow for the optimization of deep neural networks [123, 124].

Indeed, studying the capabilities of deep learning, Lin, Tegmark, and Rolnick recently argued that tensor network models may provide an explanation as to the good performance of deep neural networks [129]. Although it has been known for long that mathematical theorems guarantee that very wide neural networks are universal approximators, i.e. are able to learn arbitrary functions arbitrarily well, they observe that functions of practical interest can often be learned more “cheaply” by hierarchical networks with comparatively narrow layers. They argue that this is due to characteristics such as symmetry, local structure, compositions, and polynomial combinations of log-probabilities which frequently occur in physics and that deep networks capture such aspects in a manner similar to tensor network models in quantum information processing. Indeed, they prove several “no-flattening theorems” which establish that certain deep neural networks cannot be approximated by shallow ones without loss of accuracy and efficiency. Of particular interest in the context of quantum machine learning is the authors’ result that a shallow neural network cannot multiply n variables using fewer than 2^n neurons in its single hidden layer.

In an important practical contribution, Stoudenmire and Schwab assumed a tensor network point of view on deep learning [130]. They showed how algorithms for the optimization of tensor networks can be adapted to supervised learning tasks. In particular, they work with matrix product states to parameterize deep networks for image classification. On the MNIST data, a standard benchmark data set in machine learning, they observed less than 1% test error and thus reached state of the art performance.

Results like this spawned further research into tensor network inspired deep learning. For instance, Glasser, Pancotti, and Ciracet recently explored the connection to probabilistic graphical models, a venerable class of machine learning models [131]. They considered generalized tensor networks where information from a tensor might be copied and reused in other parts of the network, showed how to integrate this idea into common deep learning architecture, and derived an algorithm to train such networks in a supervised manner. This proved to overcome the limitations of regular tensor networks in higher dimensions while not losing computation efficiency. In experimental evaluations with image- and sound data, they found their method to improve on previously proposed tensor network algorithms. However, an observation most interesting in the context of quantum machine learning is that their approach can, in principle, be implemented on quantum computers and may therefore impact future research on quantum assisted machine learning.

Tensor networks therefore constitute interesting models in the intersection of the disciplines of quantum computing and machine learning and there is a quickly growing scientific community dedicated to the topic. For instance, several workshops on tensor networks for machine learning have already been held at the NeurIPS conference, a primary venue of machine learning research. However, so far, corresponding methods have not yet widely caught on and not yet found their way into mainstream machine learning technology. What is noticeable, though, is that researchers from Google, the Perimeter Institute, and the company X Development LLC recently released TensorNetwork, an open source library for implementing tensor network algorithms [132]. This library is supposed to support physicists and machine learners alike and provides functionalities for efficient high volume sparse data handling and tensor factorization. Whether or not initiatives like this will boost the use of tensor networks in theory and practice just as other libraries did in the case of deep learning technology remains to be seen.

5.1.3 Digital Annealing

Adiabatic quantum computers such as produced by D-Wave realize an energy minimization process called quantum annealing and they are specifically tailored towards solving quadratic unconstrained binary optimization problems (QUBOs) of the general form in equation (18) in section 3.1. Specific instances of QUBOs occur in the context of verification-, planning-, or assignment problems in areas such as, say, logistics or finance. While QUBOs therefore are of considerable practical importance, they are also difficult to solve in general because they pose combinatorial optimization problems which are NP-hard in general. It is therefore expected that (adiabatic) quantum computing will have economic impact as it provides novel approaches towards industrially relevant problems.

Alas, the technical effort required for running present day adiabatic quantum computers is substantial. While they have the potential to deliver unprecedented computing power, they must be maintained at temperatures near absolute zero and be protected against magnetic interference, thermal variation, and mechanical vibration in order for their physical realizations of logical qubits to remain in superposition. Just the amount of energy required to maintain a reliable cryogenic environment for a current D-Wave 2000Q machine is estimated to exceed 25 kWh [133]. Present day adiabatic quantum computing is thus an expensive endeavor and may not yet amortize its costs in industrial applications.

However, in machine learning, QUBOs as in (18) are known as Hopfield energy minimization problems and play a central role in the theory of Hopfield networks [71]. Hopfield networks are a special kind of neural networks inspired by physical representations of spin glass phenomena [70] and they are of considerable theoretical interest because they provide simple models for higher cognitive processes such as memory retrieval. Hopfield networks have a venerable history and are established textbook material [22]. Among others, there exist various classical algorithms for Hopfield energy minimization ranging from random updates, over greedy gradient descent methods to simulated- and mean field annealing.

Against this backdrop, Fujitsu has developed special purpose digital hardware for solving QUBOs. They refer to their technology as a *digital annealer* and market it as a solution that rivals the utility of quantum computers [134].

Fujitsu's system is based on conventional complementary metal-oxide-semiconductor (CMOS) technology and von Neumann architectures; their dedicated chip fits onto a single circuit board, works at room temperatures, and does not require a complex support infrastructure. The optimization algorithm implemented on this hardware is based on simulated annealing which is extended in several directions. For instance, the method employs a parallel trial Monte Carlo procedure which considers several switches of decision variables (simulated qubits) in parallel. This is supposed to overcome problems with respect to low acceptance probabilities in common annealing algorithms. The method also involves a so-called dynamic offset mechanism which raises acceptance probabilities after those iterations in which the Monte Carlo scheme did not make any progress. This is empirically observed to help the algorithm overcome narrow barriers in the energy landscape of the problem to be solved, and thus to make faster progress towards solutions. In its current version, the system can solve QUBOs with up to 1024 variables [135].

A similar digital annealing technology has been independently developed by a team of researchers at TU Dortmund and Fraunhofer IAIS [133, 136]. Their system is implemented on very affordable field programmable gate arrays (FPGAs). It can currently solve QUBOs of up to 2048 variables and thus exceeds the number of variables of the D-Wave 2000Q adiabatic quantum computer at a mere 0.006% of its power consumption. Moreover, while D-Wave machines can currently only solve fully connected problems with up to 119 variables, the FPGA-based solution supports dense parameter matrices for all 2048 variables.

The system has been shown to successfully solve machine learning problems such as k-means clustering, maximum a-posterior prediction, or binary support vector machine training. The optimization algorithm running on this hardware takes parallelism into account and is based on a customizable $(\mu + \lambda)$ evolutionary algorithm which allows for tuning the maximal problem dimension n, the number of parent solutions μ, the number of offspring solutions λ, and the number of bits per coefficient Q_{ij}. When working with low-budget FPGAs, this makes it possible to allocate more FPGA resources either for parallel computation (parameters μ and λ) or for the problem size (parameter n and bit depth of Q_{ij}).

5.1.4 Quantum Inspired Data Clustering

The term quantum clustering refers to a class of quantum mechanically inspired density-based clustering algorithms which assume that clusters are defined in terms of regions of more densely distributed data points. Specifically, quantum clustering algorithms model a given set of data in terms of a Gaussian mixture model consisting of one mixture component per data point. This Gaussian mixture is then considered as a quantum mechanical wave function for the data set and a quantum potential is constructed such that the data wave function becomes a stable solution to the time-independent Schrödinger equation. Gradient descent on this potential causes data points to move towards nearby local minima and data points that end up close to one another are assumed to belong to the same cluster [137].

More elaborate versions of this approach replace gradient descent by quantum evolution which can be understood as a kind of non-local gradient descent that, in turn, is capable of tunneling through potential barriers. While this has considerably higher computational costs, it allows for interesting data visualizations and the identification of substructures turning the method into a hierarchical clustering approach [138].

5.1.5 Quantum Inspired Gravitational Search

Gravitational search algorithms (GSAs) are among the newest in the class of swarm optimization algorithms which rely on the metaphor of gravitational interaction between data objects [139]. GSAs have the advantage that they are easy to implement and capable of escaping from local optima of an objective function. They have by now become established tools for effective and efficient global optimization in solving various kinds of continuous problems. In particular, the binary version, BGSA, applies to solving binary encoded problems and the discrete version, DGSA, allows for solving combinatorial problems [139].

Nezamabadi-pour [139] introduced a population based meta-heuristic search algorithm that is a binary quantum inspired gravitational search algorithm (BQIGSA) combining both gravitational search and quantum computing. His underlying idea is to solve binary encoded problems by a quantum bit superposition together with a modified rotation Q-gates strategy. Nezamabadi-pour evaluated the algorithm's effectiveness by performing experiments on combinatorial 0–1 knapsack problems or Max-ones functions. He found that BQIGSA can compete with classical BGSA, conventional genetic algorithms, binary particle swarm optimization including a modified version, a binary differential evolution, a quantum inspired particle swarm optimization, and three well known quantum inspired evolutionary algorithms.

Lou et al. [140] provide a concrete use case for quantum inspired binary gravitational search algorithms, namely the problem of predicting failure times of cloud services. They argue the importance of this case by pointing out that data centers coordinate several hundred thousand heterogeneous tasks to provide the services' high reliability. The authors motivate their research by the great challenge to acquire the optimal parameters for a relevance vector machine approach towards solving nonlinear predicting problems. In practical evaluations, they observed similar to better predicting performance of their IQBGSA-RVM algorithm compared to the baselines of chaotic genetic algorithms, binary gravitational search algorithms, binary particle swarm optimization, quantum inspired binary particle swarm optimization and standard BQIGSA (which all employ relevance vector machines).

5.2 Machine Learning for Quantum Computing

The notion of machine learning *for* quantum computing commonly refers either to the use of classical methods for preparing inputs for quantum processing units and processing outputs obtained from quantum hardware, or to the use of classical methods for designing quantum circuits.

5.2.1 Quantum Circuit Design

Quantum circuit design for quantum gate computing requires considerable experience, a deep understanding of the mathematics of quantum information processing, as well as a good deal of creativity. Because quantum circuit design can thus be considered a difficult task [89], the idea of using machine learning or optimization algorithms for this purpose arose early on and can be traced back to the 1990s.

For instance, Rubinstein [141] is concerned with evolutionary algorithms for quantum circuit design. He discusses possible encoding schemes and fitness functions and presents evolved circuits that allow for the production of entangled states. Leier [142] also works with genetic algorithms for quantum circuit design, yet focuses on how to address exponentially large search spaces for operators on exponentially large quantum states. His almost twenty year old findings suggest that quantum circuits can be evolved to a certain extent or that human experts can manually infer suitable quantum circuits from evolved solutions for small problem instances. Notably, he observed that the use of evolutionary crossover operators seemed to deteriorate the quality of the solutions found.

Now that working prototypes of quantum computers have become available, design procedures similar to the above can be tested in practice. For instance, Franken et al. [143] are concerned with variational quantum eigensolvers (VQEs). These are hybrid algorithms that combine classical and quantum computing steps in order to determine the eigenvalues of large matrices. Solutions to this general problem are sought after in many areas of science and engineering. For instance, in quantum simulations, the matrix in question often is the Hamiltonian of a quantum system and its lowest eigenvalue is of interest as it characterizes the ground state of the simulated system.

Using the VQE approach, a quantum subroutine is run inside of a classical iterative optimizer. This quantum subroutine prepares a state based on a set of given parameters and performs a series of measurements in the appropriate basis. Measurement results are read into a classical memory and are then used to classically estimate expected values of the parameters for the next iteration. Franken et al. observe that the efforts for estimating gradients of the kind of cost function that occur in this process are considerable. As a remedy, they therefore work with a weight-agnostic evolutionary scheme. They test their approach on real quantum hardware in the IBM quantum experience environment and use the automatically determined circuits to solve benchmark problems such as the transverse field Ising Hamiltonian and the Sherrington-Kirkpatrick spin model [144].

Indeed, the idea of parameterized quantum circuits has lately attracted increasing attention. It allows for the use of reinforcement learning [145, 146, 147] and other learning algorithms [148, 149] for quantum circuit design. Similarly, machine learning methods are increasingly considered as tools for solving quantum circuit mapping problems, i.e. problems of mapping a given general quantum circuit design onto a specific NISQ architecture [150, 151, 152, 153].

5.2.2 Quantum State Preparation

Machine learning approaches are more and more considered for other applications beside automatic circuit design. For example, they are used as a tool for preparing the initial state and for modelling the noise characteristics of a quantum computing system.

As will be detail below, the ability to load classical data efficiently into quantum states is the basis for the realization of many quantum algorithms. However, the best known general methods require $\mathcal{O}(2^n)$ gates to load an exact representation of a generic data structure into an n-qubit state. Therefore, scaling issues need to be taken into account because scaling can easily predominate the complexity of a quantum algorithm and thereby impair any potential quantum advantage.

Grover and Rudolph demonstrate how log-concave and other efficiently integrable probability distributions can be approximately encoded into an m-qubit register [154]. Even though the method in Grover and Rudolphs seminal note is presented for the univariate case, they explain that an extension to multiple dimensions, as it is usually the case in machine learning, is possible. Nevertheless, the described approach does require that the underlying distribution can be efficiently integrated by a classical algorithm. This precondition is frequently violated for most high-dimensional distributions.

Zoufal et al. [155] present a hybrid quantum-classical algorithm for efficient, approximate quantum state loading as an alternative to the Grover and Rudolph Method. For this purpose, they describe quantum generative adversarial networks (qGANs) to learn a generative model that prepares the desired probability distribution. For this, a probability distribution is given implicitly via data samples. The qGAN is composed of two components: the generator, which is a parametrizable quantum circuit, and the discriminator, a classical neural network. Measuring the result of the quantum circuit corresponds to a sample from the distribution that is induced by the circuit. The parameters of the classical network are tuned to distinguish these generated samples from the true data samples. Based on this classification, the parameters of the quantum circuit are tuned to output to improve the quality of the learned distribution. During training, the qGAN learns a low-dimensional representation of the probability distribution underlying the data samples and loads it into a quantum state. The loading requires $\mathcal{O}(\text{poly}\, n)$ gates and can thus enable the use of quantum algorithms, such as quantum amplitude estimation, which require to load a specific distribution. The idea of qGAN distribution learning and its loading method have been shown to work in simulations as well as in actual implementations on superconducting quantum processors.

5.2.3 Quantum Noise Modelling

Harper et al. [156] focuses on the output distribution that is observed through measurements and the noise that affects the accuracy of those measurements, instead of manipulating the input distribution.

When building large-scale quantum computers, Noise arising from various sources is the main obstacle. Quantum systems with sufficiently uncorrelated and weak noise are required for solving large real-world problem instances. Even though there has been substantial progress in designing hardware specific error correcting codes, as well as improved measurement and qubit hardware, continued progress depends on the ability to characterize quantum noise faithfully and efficiently with high precision [157].

In [156], the parameters of a classical probabilistic graphical model are tuned such that the model reproduces the noise distribution of a superconducting quantum processor. The model puts out an estimate of the effective noise and can detect correlations within arbitrary sets of qubits. It can also be applied to understand how the noise between pairs of qubits correlates. Visualization can be generated to discover long-range dependencies between the noise of specific qubits. In fact, the method revealed previously unknown error correlations within the device that was used for the experimental evaluation. The method is the first implementation of a provably rigorous and comprehensive diagnostic protocol capable of being run on real-world quantum processors, according to the authors. It builds the foundation for calibration in the presence of cross-talk, bespoke quantum error-correcting codes, and customized fault-tolerance protocols that can greatly reduce the overhead in a quantum computation.

5.3 Quantum Enhanced Machine Learning

From the point of view of machine learning practitioners, the idea of quantum enhanced machine learning is arguably among the most exciting aspects in the broad field of quantum machine learning. It deals with the use of quantum computing algorithms for solving computationally demanding learning tasks, i.e. with the processing of classical data on quantum devices to realize intelligent systems.

The expectation is that quantum speedup will make it possible to considerably accelerate learning processes or even tackle problems which are still beyond reach even for current super computers. Indeed, many common learning tasks involve linear algebra routines on very large systems of equations or optimization or search problems for which it seems likely that quantum advantages can be realized.

Since worldwide research on quantum enhanced machine learning has noticeably intensified over the past couple of years, the literature has already become vast, and numerous quantum methods and algorithms have recently been reported (see, for instance, the introductory papers by Dunjiko et al. or Biamonte et al. [46, 47] and the references therein). The following survey will thus begin with a broad overview and then review quantum algorithms for several specific machine learning problems where quantum solutions appear to be auspicious.

Aïmeur et al. [158] were among the first to address quantum information processing (QIP) for machine learning. They review QIP concepts and investigate novel learning tasks that run within environments where information is fundamentally quantum mechanical. They illustrate their idea by using the case of quantum data set clustering and providing examples of possible quantum clustering algorithms.

Riste et al. [159] specifically treat problems for which there exists a proven quantum advantage. That is, they consider rather didactic learning problems whose solution is expensive on classical computers but can be efficiently obtained on quantum computers. According to the authors, most such problems involve the repeated use of an *oracle*.

At this point it seems warranted to clarify the notion of oracles because many quantum computing algorithms posit their existence. We therefore note that, while it is often difficult to determine a solution to a given problem, it is often also simple to verify if an alleged solution really solves the problem at hand. Consider for instance the problem of prime factorization. While it requires some effort to determine that $1, 2, 3$ and 7 are prime factors of 42, it is comparatively easy to verify that they are. By the same token, it is also easy to verify that, say, $1, 5$ and 13 are not the prime factors of 42. A function that easily accomplishes such a verification for a given problem is called an oracle. Just for completeness we also note that for many if not most computational intelligence problems oracles do not exist. For instance, in chess it is generally difficult (if not impossible) to determine the single best move for a given game state. Contrary to prime factorization, it will generally be also at least as difficult to verify if an alleged best move is indeed optimal.

Riste et al. observe that the costs incurred by an oracle-base quantum algorithm can be determined by its querying complexity, i.e. by the number of oracle calls needed to find a solution with a given probability. While showcases of oracle-based quantum algorithms have already been verified experimentally on various platforms with different physical realizations of logical qubits, the authors note that these showcases typically involved problems that could be modeled in terms of very few qubits. Based on toy scenarios like these, however, it is hard to argue for quantum supremacy because classical algorithms, too, could solve problems like these with a few queries to an oracle. The authors therefore investigated the performance of classical and quantum approaches in solving a more demanding oracle-based problem known as *learning parity with noise*. Using a custom made five-qubit superconducting processor (consisting of superconducting qubits) as well as classical Bayesian computing, they observed a query complexity in favor of quantum processing which substantially increases depending on the acceptable error rate and the problem size. Their important major finding is thus that a significant quantum advantage can already arise even in present day noise-intensive systems.

While general results such as these are encouraging, one has to keep in mind that current NISQ devices can only realize few logical qubits and still suffer from limited coherence times. That is, even if the findings in [159] suggest that limited fault tolerance may not be the most pressing technical challenge in quantum machine learning, certain quantum learning algorithms may still not be practical. O'Quinn and Mao [160] emphasize this and point out that existing platforms do not yet allow for practical implementations of some of the more fanciful ideas discussed in the quantum machine learning literature.

5.3.1 Quantum Algorithms for Linear Algebra

One of the reasons for why machine learning models often assume input data to be encoded in terms of data points $\boldsymbol{x} \in \mathbb{R}^m$ is rather prosaic and due to technological circumstances: as long as the dimension m of a given vector is not too large, conventional digital computers can easily perform linear algebraic operations. Indeed, there exist venerable software libraries such as BLAS and LAPACK which date back to the 1970s, compile on almost any operating system, and allow for highly efficient linear algebra computations. Moreover, modern graphics processing units (GPUs) allow for rapid, parallel memory access and are specifically designed to facilitate matrix vector operations. In a sense, numerical computing technology and machine learning co-evolved, and it is therefore not surprising that many machine learning algorithms involve matrix vector products, matrix inversion, or the decomposition of matrices into factors of lower rank.

Against this backdrop it seems reasonable to attempt to harness potential quantum advantages for machine learning sub-routines since quantum computing is all about matrices of size $2^n \times 2^n$ acting on 2^n dimensional vectors.

Harrow, Hassidim, and Lloyd [119] developed a quantum algorithm for solving systems of linear equation which can be written in terms of sparse and well conditioned matrices. This algorithm is considered to be one of the more fundamental, presently known methods with provable quantum speedup over their classical counterparts. It thus falls within the same category as Shor's factoring algorithm (based on Coppersmith's quantum Fourier transformation [161]), Grover's search algorithm, or quantum random walk algorithms [162, 163] and is by now simply known as the HHL algorithm.

Harrow, Hassidim, and Lloyd are specifically concerned with solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x}$ where the $n \times n$ matrix $\boldsymbol{A}$ is Hermitian and $\boldsymbol{b}$ is an n-dimensional unit vector. They then suppose an amplitude encoding $|b\rangle$ of $\boldsymbol{b}$ (for instance using the method in [164]) and a Hamiltonian operator exp$(i\boldsymbol{A}t)$ and perform quantum phase estimation [113, 114], ancilla bit rotation, inverse quantum phase estimation, and measurement to obtain an estimate of the eigenbasis of the coefficient matrix in which it is easy to invert it.

If the coefficient matrix of the considered linear system of n variables is sparse (not densely populated) and has a low condition number k and—crucially—if users are mainly interested in the result of a measurement on the solution vector rather than in the solution vector itself, then HHL has a runtime of $\mathcal{O}(\log n \cdot k^2)$. This constitutes an exponential speedup over the fastest classical algorithms which generally run in $\mathcal{O}(n \cdot k)$. Importantly, improvements over the originally proposed approach are possible if phase estimation is replaced by the "linear-combination of unitary method" which approximates matrix inverses via Fourier- or Chebyshev-series expansions [165]. Moreover, since the HHL algorithms has been verified experimentally, it is known to work on existing quantum computers, at least for small problems which meet its prerequisites [166, 167, 168].

The singular value decomposition, also dubbed a "singularly valuable decomposition" [169], is one of the most important matrix factorization techniques with numerous practical applications in science and engineering. Assuming that a universal quantum computer with access to a QRAM was available, Rebentrost et al. [170] present a quantum singular value decomposition which is exponentially faster than its classical counterparts. Making less wide reaching assumptions, Gilyen et al. [117] propose a quantum singular value "transformation" algorithm that is based on the idea of qubitization [171] and computes a bounded polynomial approximation of the singular value decomposition. Note that the term qubitization refers to a technique for simulating the time evolution operator exp$(i\boldsymbol{A}t)$ that features prominently in the HHL algorithm. For this, Low et al. [171] prove that it can be done with low error using comparatively simple quantum random walk circuits.

Gilyen et al. observe that their model leads to rather simple quantum circuits which tend to involve only constantly many ancilla qubits. From a general application point of view, their method allows for estimating Moore-Penrose pseudo-inverses and thus applies to regression problems. Moreover, the authors also observe that their quantum singular value transformation leads to a unified framework for several quantum algorithms. Methods that can be seen as special cases of their approach include fixed-point amplitude amplification, robust oblivious amplitude amplification, and certain quantum walks. Yet, overall the work in [117] is of mostly theoretical nature; attempts of a practical implementation are not reported even though the authors provide a detailed discussion of how to implement quantum linear algebra methods by representing matrices in terms of unitary circuits and vectors in terms of quantum states.

Clader, Jacobs and Sprouse [172] generalize the HHL algorithm. They propose a state preparation routine which is capable of initializing generic states and they integrate a quantum compatible preconditioner, which enlarges the problem space for which solutions can be obtained with an exponential speedup over classical linear systems solvers. The authors verified their algorithm's applicability by letting it compute the electromagnetic scattering cross section of an arbitrary target. As a result, it was found to be exponentially faster than the best classical algorithm.

Huang, Bharti and Rebentrost [173] deal with near-term quantum algorithms for linear equation systems of the form $\boldsymbol{Ax} = \boldsymbol{b}$ and examine the use of variational algorithms. As part of their research, the authors develop a near-term algorithm, which is founded on the classical combination of quantum states (CQS) method. They conducted experiments of solving large linear systems by simulating the quantum algorithm on a classical computer and report the approach to work well.

Subasi, Somma and Orsucci [174] introduce two quantum algorithms based on evolution randomization, a simple variant of adiabatic quantum computing, to prepare a quantum state $|x\rangle$, which is proportional to the solution of the linear equation system $\boldsymbol{Ax} = \boldsymbol{b}$. Both algorithms are easy to implement and designed using families of Hamiltonians that are linear combinations of products of $\boldsymbol{A}$. A quantum oracle is supposed to be given that, for any row of $\boldsymbol{A}$, outputs the nonzero matrix elements and their indices. Given this prerequisite, their algorithms exhibits an exponential quantum speedup.

To solve a system of linear equations, Lee, Joo and et al. [175] introduce a hybrid quantum algorithm built on the HHL algorithm. It reduces the circuit depth from the original algorithm without accuracy loss of the results. On the contrary, the authors' experiments (using 4 qubits by IBM Quantum Experience) produced better results, that is a higher accurate performance on specific systems of linear equations.

Bravo-Prieto et al. [176] take as a starting point the limitation that existing quantum solvers of linear equation systems are hardly implementable due to the required circuit depth. The authors introduce a variational quantum linear solver (VQLS), which is a hybrid algorithm designed for near-term quantum computers. This solver seeks to variationally prepare $|x\rangle$ such that $A|x\rangle \propto |b\rangle$ and Bravo-Prieto et al. then derive a termination condition guarantying the achievement to a desired precision. The authors verified their algorithm using Rigetti's quantum computer.

Xu et al. [177] take up the challenge of a quantum algorithm's high demands on the circuits depth (leading to a high demand on the quantum device's universal fault-tolerance) to solve linear algebra tasks. The authors developed variational algorithms, which are compatible with noisy intermediate-scale quantum devices. They demonstrate that the linear equation system's solutions and matrix-vector multiplications are translatable as the ground states of the constructed Hamiltonians. They implemented their algorithm using the IBM quantum cloud services and observed a high solution fidelity of 99.95%.

5.3.2 Quantum Algorithms for Regression

The term regression analysis refers to a broad spectrum of statistical methods for estimating relationships between a dependent variable and one or more independent variables. While there exist many different algorithms for fitting a regression model to data, the arguably most well known approach is least squares regression. At its heart, this is a simple linear technique which nevertheless allows for dealing with non-linear models. The basic setting is as follows: Given a data matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ whose columns represent (possibly non-linearly transformed) data points and a target vector $\boldsymbol{y} \in \mathbb{R}^n$, the task is to identify a weight vector $\boldsymbol{w} \in \mathbb{R}^n$ such that the loss $\|\boldsymbol{X}^\intercal \boldsymbol{w} - \boldsymbol{y}\|^2$ is minimal. This can be understood as an almost trivial supervised learning problem since its unique closed form solution is known to be given by $\boldsymbol{w} = (\boldsymbol{X}\boldsymbol{X}^\intercal)^{-1}\boldsymbol{X}\boldsymbol{y}$. Moreover, for the practical computation of this expression there exist fast and numerically stable gradient descent schemes. Nevertheless, because least squares regression is such a fundamental technique in data science and machine learning, it has attracted the attention of quantum computing researchers.

Initially, work in this direction was mainly theoretical. For instance, assuming the existence of a universal quantum computer which when queried about an element in a given row of a sparse matrix $\boldsymbol{X}$, produces a quantum state that encodes the column number and, moreover, is capable of producing copies of an input state $|y\rangle$, Wiebe, Braun, and Lloyd [178] present an HHL-based algorithm that estimates $|w\rangle$ up to scale.

Schuld, Sinayskiy, and Petruccione [179] address certain shortcomings of this approach. In particular, they extend it towards dense matrices and improve on the dependency on the condition number. Their idea is to create amplitude encoded quantum states that represent the singular value decomposition of the data matrix and then to run phase estimation to invert the unknown singular values. They acknowledge that the problem of preparing quantum states which encode classical data is non-trivial and assume that states are either given in a quantum random access memory (QRAM) from which the algorithm could access them or that there is an oracle that loads data into an entangled register.

Some of the computational steps in [179] apply ideas which were first brought forth by Wang [180] who presents a quantum algorithm for fitting a linear regression model and focuses on estimating the quality of the fit and on assessing whether the given data set is suitable for quantum regression at all. This allows his algorithm to output optimal parameters in classical form. However, Wang, too assumes availability of a QRAM which might not be technically feasible in the foreseeable future.

Yu et al [181] introduce an improved quantum algorithm for ridge regression, a regularized version of ordinary least squares regression. Ridge regression allows for more robust estimates but comes with the need of having to estimate additional hyper-parameters. The authors develop a quantum algorithm for ridge regression that utilizes the parallel Hamiltonian simulation technique for simulating Hermitian matrices in parallel and for implementing a quantum k-fold cross-validation that estimates the ridge regression's predictive performance. Yu et al. compose their algorithm in two stages: Stage 1 searches for a suitable hyper-parameter α that optimizes the predictive performance of the regression model by running quantum cross-validation. Stage 2 creates a quantum state which encodes the optimal fitting parameters of ridge regression with such an α, that is later used to predict new data. Again, a major (practical) weakness of the proposed method is that the authors quantum encoded data to be available in a QRAM.

Hou et al. [182] make such an assumption, too. The authors propose a quantum partial least squares regression algorithm. Partial least squares regression is yet another variant of the ordinary least squares approach which allows for addressing multiple correlation problems. The authors develop a method for quantum eigenvector search in order to accelerate the regression parameter selection and propose a density matrix product to avoid multiple QRAM queries for constructing residual matrices. If a QRAM were available, this algorithm would run exponentially faster than classical partial least squares estimation techniques.

Liu and Zhang [183] further improve on techniques such as the above. To this end, their algorithm takes statistic leverage scores of a data matrix into account. They show how to generate a quantum state proportional to $|w\rangle$ in $\mathcal{O}(\log n)$ time for sparse and well-conditioned $\boldsymbol{X}$. Again, they assume efficient quantum access to classical data was possible.

Gilyen, Song, and Tang [184] rightfully point out that many QML algorithms for low-rank regression require input to be stored in a QRAM especially those that theoretically achieve runtimes that are poly-logarithmic in the dimension of the data under consideration. As a remedy, they describe a classical algorithm for linear regression that borrows from the HHL algorithm and improves on the method due to Chia et al. [116]. Their stochastic gradient algorithm exploits recent numerical sparsity techniques for fast eigenvector computation and regression analysis and though not designed as a quantum algorithm point towards possible genuine quantum solutions.

Finally, Date and Potok [185] recently showed how to perform linear regression on adiabatic quantum computers. In order to accomplish this, they cast the regression problem as a quadratic unconstrained binary optimization problem which they then solve on a D-Wave 2000Q adiabatic quantum computer. While this approach does not suffer from any (as of yet technically unrealistic) assumptions as to data encoding, it incurs a certain loss in numerical precision. However, the authors compare their results obtained on a D-Wave machine to those resulting from classical numerical solution implemented in Python running on desktop computer equipped with a current multi-core Intel processor. Their experiments reveal that the adiabatic quantum implementation achieves up to 2.8 fold speedup over the digital implementation but is on par with respect to the regression error metric.

5.3.3 Quantum Algorithms for Clustering

Prototype-based clustering is an unsupervised machine learning problem where minimization of a loss function allows for partitioning a set of n data points into $k \ll n$ clusters which are defined in terms of representative elements. The arguably most well known instance of a prototype-based clustering method is k-means clustering where the n data points $\boldsymbol{x}_j \in \mathbb{R}^m$ are Euclidean vectors and the k prototypes $\boldsymbol{\mu}_i \in \mathbb{R}^m$ are the centroids of their respective clusters. These are typically found through an iterative minimization of a variance-based loss; alas, while this loss is commonly considered to be intuitive, its minimization actually poses an NP-hard problem even for the simple case where $k = 2$ [186]. Popular classical k-means clustering algorithms such as those due to Lloyd, MacQueen, or Hartigan are therefore but heuristic approaches to the problem and there is no guarantee that they will determine the optimal solution.

However, for $k = 2$, k-means clustering becomes a bipartition problem that can be cast as a QUBO and thus be solved via adiabatic quantum computing [73]. The corresponding Hamiltonians are derived from a reformulation of the conventional k-means objective which is based on Fisher's analysis of variance and leads to an equivalent criterion that also allows for kernel k-means clustering via adiabatic quantum computing [187]. In both cases, the adiabatic quantum optimization procedure performs an exhaustive search over the space of all possible solutions and does so quadratically faster than classically possible. While the validity of the ideas in [73, 187] was originally verified in simulation experiments, the approaches have meanwhile been implemented on D-Wave computers and were found to be practically feasible [188, 189, 190].

If the QUBO in [73] is modified appropriately, the approach also allows for the estimation of k-medoids, i.e. cluster prototypes that approximate cluster means [191] and the resulting algorithm has been successfully applied to solve cardinality-constrained index-tracking problems in the financial industry [192].

The work in [73] also shows how to extend adiabatic quantum bipartition clustering towards non-numeric data for which one can define similarity- or distance measures. Given such measures, relations among non-numeric data points (such as text strings) can be modeled in a graph and the bipartition clustering problem becomes a minimum graph cut problem. Hamiltonians for this general problem class are algebraically identical to those for $k = 2$-means clustering and the approach is practically viable.

A remaining practical challenge with these kind of quantum algorithms for bipartition clustering is that they require as many logical qubits as there are data points to be clustered. While this is no different on digital computers, quantum computers that could manipulate as many logical qubits as modern, large scale data sets would require, do not yet exist.

Aïmeur, Brassard, and Gambs [193, 194] discuss how quantum gate computing can speed up unsupervised learning algorithms used in data clustering. Specifically, they propose quantum versions for minimum spanning tree clustering, divisive clustering, and k-medians clustering. The basic idea is to consider an oracle which provides knowledge as to distances between data points and to use this oracle in algorithms based on Grover's phase amplification.

Tomesh et al. [195] observe that quantum machine learning often assumes classical data to be encoded in term of quantum states in superposition which, due to the difficulty of known encoding schemes, can annihilate potential quantum speedup. They therefore investigate a coreset-based approach towards prototype-based clustering with reduced loading overhead. Coresets are an important concept in machine learning as they can allow for approximating large sets of data points in terms of small representative sets that can quickly be determined (often in linear time). Given a coreset for a problem under consideration, Tomesh et al. apply the quantum approximate optimization algorithm (QAOA) [84] and run numerical simulations to compare the their approach against classical k-means clustering. Their results indicate that there exist data sets where QAOA might outperform standard k-means clustering on a coreset. However, the authors point out that, for their method to show a quantum advantage over conventional k-means algorithms, problems have to be rather special in that they have to have appropriate coresets. This is, because their method crucially hinges on the existing of a characteristic coreset which may not exit in general.

5.3.4 Quantum Algorithms for Nearest Neighbor Search

In what follows, the term nearest neighbor search is used to refer to the following general setting: Given an unstructured set $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^m$ of prototypes and an input data points $\boldsymbol{x} \in \mathbb{R}^m$, determine which prototype is closest to the input, i.e. solve $\boldsymbol{x}^* = \operatorname{argmin}_i \|\boldsymbol{x}_i - \boldsymbol{x}\|$. Given the above discussion, it is clear that this problem occurs in the context of assigning data to clusters. Moreover, if the prototypes are labeled, nearest neighbor search also allows for nearest neighbor classification in that input data points $\boldsymbol{x}$ can be automatically labeled using the label of their respective closest prototype $\boldsymbol{x}^*$.

Classically, naïve nearest neighbor search requires efforts of $\mathcal{O}(n)$. However, there exist specific data structures such as k D-trees which allow for nearest neighbor search in $\mathcal{O}(\log n)$. This gain comes at the cost of having to pre-process the prototypes in order to structure them correspondingly. This typically requires efforts of $\mathcal{O}(n \cdot \log n)$ which amortize if the prototypes have to be searched repeatedly, for instance, in data base search scenarios. While nearest neighbor search is thus not too demanding a problem for classical computers, it is interesting to observe that quantum computing can, in principle, accomplish search in $\mathcal{O}(\sqrt{n})$ even without data pre-processing.

Wiebe et al. [196] stress the importance of nearest neighbor search in clustering and classification and present quantum nearest neighbor approaches for binary classification and k-means clustering. In essence, their approach allows for quantum computation of Euclidean distances between sparse data points and assumes the availability of quantum oracles that can efficiently look up the j-th value of prototype $\boldsymbol{x}_i$ and identify the l-th non-zero value in prototype $\boldsymbol{x}_i$. They furthermore assume that it is possible to prepare a quantum state that encodes $\|\boldsymbol{x}_i - \boldsymbol{x}\|$ and present a quantum circuit for this purpose. They then apply a classical quantum minimum search algorithm due to Dürr and Høyer [197] which itself is a variant of Grover's amplitude amplification procedure for searching with oracles [37, 38]. They prove that the number of oracle queries of their method scales as $\mathcal{O}(n \cdot \log n)$ which is thus as fast as informed classical search. In numerical simulation experiments on common machine learning benchmark data they find that their technique is robust to noise arising from coherent amplitude estimation, performs well, and asymptotically outperforms classical sampling techniques for nearest neighbor search.

Llyod et al. [198] are concerned with the problem of k-nearest neighbor search. Whereas naïve classical solutions to this problem would require efforts of $\mathcal{O}(kn)$, the quantum algorithm proposed in [198] has a runtime complexity of only $\mathcal{O}(\log kn)$ and thus achieves exponential speedup. However, the authors once again assume that classical data has efficiently been loaded into a QRAM so that their algorithm can access it in a quantum parallel manner. They then proceed to prepare states that represent sub-norms of the input data on which they run adiabatic quantum optimization to identify and select the k nearest prototypes to a given query.

In a recent contribution, Basheer et al. [199] integrate ideas from [196] and [198] with recent techniques for preparing data into amplitudes of quantum states [200] and present specific quantum circuits for k-nearest neighbor search in classification. Their algorithm has a runtime complexity of $\mathcal{O}(\sqrt{kn})$ but, crucially, does not make any assumptions as to the availability of QRAMs. In numerical experiments, in which the authors simulate classification problems with few classes using only few qubits, they observe the approach to work well.

They also point out that their methods seamlessly apply to downstream processing in quantum computing settings. That is, the proposed algorithm can be directly used on quantum data and thus circumvent the need of reconstructing quantum states using measurements on an ensemble of identical states in physics applications.

Concerned with a much simplified nearest neighbor setting, Schuld et al. [201] present a distance-based classifier which they realize in terms of a simple quantum interference circuit. Excluding a simple state preparation procedure, the circuit consists of only one Hadamard gate and two single-qubit measurements units. It computes distances between input data points and two prototypes in quantum parallel and has been implemented in the IBM quantum experience environment.

5.3.5 Quantum Algorithms for Classification

Pattern recognition or classification is the problem of assigning labels (signifying classes or categories) to observations of objects (represented in terms of data points) and a function or an algorithm that accomplishes classification, especially in a specific scenario, is called a classifier.

If a classifier can classify objects from two classes (say pictures of cats and dogs), it is said to be a binary classifier. A classifier that can classify multiple classes (say pictures of cats, dogs, mice, horses, elephants, ...) is called a multi-class classifier. Since any multi-class classification problem can be addressed by several one-versus-all classifiers, binary classification is the more fundamental task and the problem of training a robust and reliable binary classifier using labeled training data is one the most important problems in machine learning.

There exist numerous possible models that allow for binary classification. Some of these are probabilistic (naïve Bayes classifiers or Bayesian networks), some operate on categorical data (decision trees), and some are geometric or linear algebraic in nature (least squares classifiers, linear discriminant classifiers, support vector machines, etc.).

A simple binary linear classifier for input $\boldsymbol{x} \in \mathbb{R}^m$ is a threshold function of the following form

$$y = f(\boldsymbol{x} \mid \boldsymbol{\theta}) = \text{sign}(\boldsymbol{x}^\intercal \boldsymbol{w} - b) \tag{34}$$

whose parameters $\boldsymbol{\theta} = \{\boldsymbol{w}, b\}$ are a projection- or weight vector $\boldsymbol{w} \in \mathbb{R}^m$ and a threshold or bias value $b \in \mathbb{R}$. While the internal computations $\boldsymbol{x}^\intercal \boldsymbol{w} - b$ of this model are indeed linear, the function sign is said to be a non-linear activation function which produces outputs in $\{-1, +1\}$ and thus allows for binary decision making. It is also common to consider activation functions such as the hyperbolic tangent which produces outputs in $(-1, +1)$, the logistic function with outputs in $(0, 1)$, or rectified linear units with outputs in $[0, \infty)$. The choice of activation can impact the ease of training but the fundamental problem in each case is to determine suitable $\boldsymbol{w}$ and b. Depending on the choice of learning algorithm (least squares training, linear discriminant training, support vector training, etc.) this can be more or less challenging but is well understood in general and computationally not too demanding.

Binary linear classifiers learn linear decision boundaries which divide the underlying data space into two disjoint half-spaces, namely $\{\boldsymbol{x} \mid f(\boldsymbol{x}) \geq 0\}$ and $\{\boldsymbol{x} \mid f(\boldsymbol{x}) < 0\}$. However, it is interesting to note that many linear classification models allow for invoking the *kernel trick* [202]. This way, data can implicitly be processed in very high to infinitely dimensional kernel Hilbert spaces. This then makes it possible to use simple, linear models in order to learn highly non-linear decision boundaries.

Since linear classifiers (kernelized or not) rely on linear algebraic computations, they often allow for corresponding quantum computing algorithms. Early attempts in this direction were made by Schuld et al. [203, 204]. In [203] they argue that quantum computing can outperform classical techniques in the case of ambiguous input patterns. They develop a quantum nearest neighbor classifier that involves Trugenberger's quantum algorithm for measuring Hamming distances in quantum associative memories [205, 206]. In simulation experiments on standard benchmark data, they find their method to work well. Its runtime behavior is similar to classical implementations but the authors remark that, if there was an efficient $\mathcal{O}(n)$ approach for constructing the required quantum superposition states or if a QRAM would exist, their quantum algorithm would be independent of the number of training data, a feat that seems impossible for the classical counterpart.

In [204], Schuld et al. present a quantum perceptron, i.e. a quantum circuit that realizes the binary linear classification function in (34). They propose quantum phase encoding of $\varphi = \boldsymbol{x}^\intercal \boldsymbol{w}$ as $|\varphi\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{i\,\varphi}|1\rangle\big)$ and then use the phase estimation algorithm [113, 114] to determine the corresponding output $|y\rangle$. In terms of runtime or ease of implementation this does not seem to offer any benefits over the classical approach; however, the authors point out that their quantum perceptron yields outputs which contain quantum information. This could be used for superposition-based learning procedures in which training data enter the algorithm as a superposition of data points which would then allow for quantum parallel processing.

Wiebe et al. [207] ask if quantum computation could lead to non-trivial improvements in the computational and statistical complexity of perceptrons and present algorithms which answer both these questions affirmatively. To this end, they consider the notion of the version space of a set of training data for a binary classification problem. This term refers to the set of all hyper-planes that perfectly separate the two classes under consideration. Seen from the version space perspective, the problem of classifier training can be cast as a search problem which, in turn, can be solved using Grover's quantum search algorithm. Based on these ideas, Wiebe et al. develop two algorithms with different speedups for perceptron training. The first trains in $\mathcal{O}(\sqrt{n})$ where n denotes the number of training data points; the second trains in $\mathcal{O}\big(\rho^{-1/2}\big)$ where ρ denotes the optimal margin between the two classes under consideration. As of this writing, it is difficult to imagine how this could be achieved classically. The work in [207] therefore establishes that quantum computing can speed up perceptron training and thus improve on one of the most fundamental tasks in machine learning. The authors thus conclude that the quantum computing point of view can lead to a deeper general understanding of machine learning and may lead to models for which there is no classical analogue.

Tacchino et al. [208] have recently shown how to practically implement a simple binary-valued perceptron on an existing quantum device, namely a 5-qubits IBM quantum computer based on superconducting technology. In practical experiments, they demonstrate successful classification of 4 bits strings using 2 qubits and of 16 bits strings using 4 qubits. This hints at an exponential storage advantages over classical methods but the authors concede that generic, i.e. non binary patterns, may need quantum states or unitary transformations whose preparation requires exponentially many quantum gates. This, in turn, would eliminate the quantum advantage of their method. They also acknowledge that current NISQ era devices do not allow for arbitrary controlled operations but require them to be broken down into operations involving only single- and two-qubit gates which significantly increases circuit depth and thus the risk of decoherence. As a possible future generalization away from binary inputs, the authors point to phase encoding similar to the approach in [204].

In another recent contribution, Schuld et al. [209] observe that quantum computing bears certain similarities to the idea of working with kernel methods in machine learning in that both, quantum computers and kernel machines, process information in (possibly infinitely high-dimensional) Hilbert spaces. They argue that this can lead to new ideas for the design of quantum machine learning algorithms and, in particular, interpret the problem of encoding classical data in a quantum state as the problem of computing a non-linear kernel function that maps the data into quantum Hilbert space.

To this end, they associate a quantum Hilbert space with a machine learning feature space and derive a kernel that corresponds to the inner product of quantum states. They discuss how this allows for the encoding of computational basis states, amplitude encoding of data points, and representations of product states and use these insights to devise a parameterized circuit model of a quantum support vector machine. The circuit is composed of displacement- and phase gates whose parameters are determined via classical stochastic gradient optimization. Schuld et al. thus consider a variational quantum training algorithm that combines quantum computations with classical computations in an iterative feedback loop. Experimental verification with a simple mini benchmark data set of two-dimensional data points from two classes suggests that the idea works well in practice.

Similarly, Grant et al. [210] are concerned with the capabilities of present day NISQ devices and point out that hybrid quantum-classical algorithms where quantum computers run model circuits and classical computers perform statistical loss minimization to train the model circuit are currently the best strategy for quantum (assisted) machine learning. Building on concepts first proposed in [211], they therefore propose hierarchical quantum circuits for binary classification. In particular, they consider parameterized circuits of tensor network topology. In experiments with tree tensor networks and multi-scale entanglement renormalization (MERA) networks, they determine their parameters classically and then deploy them on an ibmqx4 machine available within the IBM quantum experience environment. The data they consider are small subsets of standard machine learning benchmarks and their results suggest that the method is robust to noise and can achieve high accuracy. Importantly, the authors stress that their method allows for classifying classical- as well as quantum data.

Finally, Date et al. [190] recently discussed how to cast the problem of training linear support vector machines for binary classification as a quadratic unconstrained binary optimization problem. While naïve classical algorithms would require efforts of $\mathcal{O}(n^3)$ for support vector machine training with n data points, setting up the QUBO requires only $\mathcal{O}(n^2)$

computations and its solution could subsequently happen on an adiabatic quantum computer. This is not demonstrated practically but, in line with their linear regression results in [185], should be entirely possible. The authors' mainly theoretical work thus suggests that well established machine learning baseline models can be trained on adiabatic quantum computers from where the resulting model parameters can easily be read into digital memories for further processing.

5.3.6 Quantum Boosting

Boosting is a machine learning technique tailored towards classifier training that is rather easy to implement and known to yield well working systems. There are several variants and flavors, but the original idea of adaptive boosting (AdaBoost) [212] is arguably still the most popular approach. Assuming a set of individually weak classifiers, AdaBoost evaluates an exponential loss to determine which (re-)weighted sum of their outputs provide a strong classifier. The resulting boosted classifier is also known as an ensemble classifier or, if the individual weak learners are non-linear threshold units such as in (34), a shallow neural network. Shortly after its inception in the 1990s, classifier boosting led to breakthroughs in rapid and reliable computer vision [213] and correspondingly trained systems were, for instance, implemented in consumer cameras to detect smiling faces. Alas, the good performance of boosted classifiers comes at the price of typically extensive training times, as it requires numerous rounds of training to select a high-performance ensemble among a very large number of individual learners. Yet, since boosting is essentially a subset selection problem, it has been identified as a setting for potential quantum speedup early on.

Neven et al. [214] propose adiabatic quantum optimization for boosting. They transform the originally continuous weight optimization problem into an optimization problem over discrete variables of low bit depth and consider an adapted quadratic loss function. This allows them to express boosted classifier training as a QUBO. In experiments with heuristic surrogates for quantum hardware as well as with an early D-Wave machine, their QBoost approach compares favorably to classical AdaBoost. It reduces training efforts and leads to better generalization and faster runtimes because the resulting ensembles are typically found to be small.

However, the authors concede that these advantages are mainly due to their bit-constrained learning model and not a quantum effect per se. The notable result is thus that a restricted (and hence implicitly regularized) model that was specifically designed for implementation on quantum hardware can outperform unrestricted conventional models. Yet, since solving the underlying discrete optimization problem might be too demanding for conventional computers if the problem size is large, the work in [214] establishes that quantum computing enables approaches towards machine learning that would be classically infeasible.

Pudenz and Lidar [215], too, consider quantum adiabatic evolution for classifier training in a manner similar to that of Neven et al. However, they also apply adiabatic quantum computing in the application phase of their system and adiabatically evolve strong classifiers identified during training on a superposition of inputs in order to be able to identify likely anomalous elements in their data space in a quantum parallel manner. The practical application they consider is the problem of verification and validation of classical software where programming errors (bugs) are the anomalies to be detected. Extensive simulation experiments indicate that their methods work well.

Schuld and Petruccione [216] consider the problem of creating a classifier ensemble to be a problem of quantum state preparation. A quantum parallel evaluation of individual quantum classifiers on such states allows for estimating their combined or aggregated decision in terms of a single qubit measurement. The authors argue that their framework allows for exponentially large ensembles and thus for advantages over classical approaches.

As a quantum gate model of a weak classifier, they consider their earlier proposal of a quantum perceptron [204] and, based thereupon, describe a protocol for preparing states that represent classifier ensembles. Crucially, this requires estimates of the accuracy of the individual classifiers which can then be used in a weighting scheme. While the authors are more concerned with the feasibility of quantum ensembles rather than with their potential advantages, they do point to several possible applications in quantum physics and remark that their ideas suggest approaches towards optimization-free quantum machine learning. In other words, a potential benefit of the proposed approach is that measuring large enough ensembles of individually weak classifiers can lead to accurate decisions without that the ensemble would have to be tuned in a training process.

5.3.7 Quantum Support Vector Machines

Kernel machines are an important class of classical machine learning models. The key idea observation is that many (linear) machine learning methods require access to the data only through inner products between feature representations of two data points, i.e. $\langle \phi(x) | \phi(y) \rangle$. The mapping $\phi : \mathcal{X} \to \mathbb{R}^d$ to the d-dimensional feature space has to be specified by the user and is crucial for the quality of the resulting model. When the dimension d is large, the explicit computation of ϕ can be circumvented via a so-called kernel function $k(x, y) = \langle \phi(x) | \phi(y) \rangle$.

The most prominent incarnation of this type of method is the support vector machine (SVM) [217]. Classically, the choice of the feature map $\phi(\cdot)$ or, more commonly, the kernel function $k(\cdot, \cdot)$ are either domain specific (for instance, string kernels, graph kernels, tree kernels, image kernels, document embedding kernels, time series kernels, ...) or generic (for instance, radial basis function kernels, polynomial kernels, Fisher kernels, ...).

In contrast, the feature map of quantum kernel machines is hardware specific. Instead of relying on problem specific knowledge to construct the feature map or the kernel, the intrinsically 2^n-dimensional Hilbert space of an n-qubit register is utilized to realize the feature map [218].

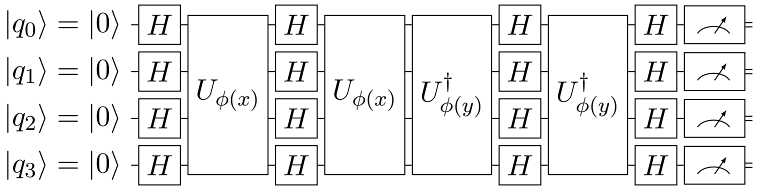

$|q_0\rangle = |0\rangle$
$|q_1\rangle = |0\rangle$
$|q_2\rangle = |0\rangle$
$|q_3\rangle = |0\rangle$
H
$U_{\phi(x)}$
$U_{\phi(x)}$
$U^\dagger_{\phi(y)}$
$U^\dagger_{\phi(y)}$

Figure 3: Variational 4-qubit circuit for quantum kernel machines. Data points x and y are both mapped into the 2^4-dimensional Hilbert space via parameters of unitary gates $U_{\phi(x)}$ and $U_{\phi(y)}$.

A schematic representation of a corresponding quantum gate circuit is shown in Fig. 3. There, a maximal superposition is prepared and then passed through n-qubit unitaries which create the feature space transformation of the data. It is important to understand that data does not enter the circuit in the discrete qubit state space. Instead, data is passed in form of parameters of universal unitary gates $U_{\phi(x)}$ and $U_{\phi(y)}$ representing (parts of) the high-dimensional feature map. Each classical n-bit binary string is interpreted as one feature and the corresponding probability amplitudes of the qubit state as feature values. The actual kernel value $k(x, y)$ is then given by estimating the transition amplitude $|\langle\phi(x), \phi(y)\rangle|^2 = |\langle 0^n|U^\dagger_{\phi(y)}U_{\phi(x)}|0^n\rangle|^2$. Clearly, the specific choice of $U_{\phi(\cdot)}$ is not fixed and can be tuned for the application at hand. Obtaining the full kernel matrix for n data points requires $n(n+1)/2$ runs of that circuit. The resulting quantum kernel matrix is then ready to be used in any kind of kernel machine, for instance, in support vector machines, kernel regression, or kernelized k-means clustering.

In addition to quantum k-nearest neighbors algorithms, Wittek [219] introduces the concepts of quantum support vector machine algorithms based on a slightly extended formulation of traditional SVMs using least-squares optimization. He indicates that quantum SVMs have a high generalization performance, even though the extension to general multi-class problems is expensive.

Wang et al. [220] take advantage of the fact that the *least squares support vector machine* algorithm shows high efficiency in processing a small quantity of data for the problem of trend recognition. To optimize the parameters of the least squares support vector machine and to establish the curve fitting model, the authors use quantum particle swarm optimization to realize a faster global parameter search. To corroborate the practicality of their ideas, the authors perform on a mathematical error analysis.

Rebentrost, Mohseni, and Lloyd [221] demonstrate that support vector machines can be implemented on a quantum computer such that they achieve logarithmic complexity in the data dimension and the number of training examples. This constitutes an exponential speedup over classical sampling algorithms which run in polynomial time. The main component of the quantum algorithm developed by the authors is a non-sparse matrix exponentiation technique which inverts the training data inner-product (kernel) matrix. However, the latter is based on quantum regression methods such as those discussed above. Since such methods assume that quantum states can either be prepared efficiently or are available for convenient lookup in a quantum random access memory, the ideas in [221] may be of limited practical feasibility on current NISQ era devices.

Havenstein, Thomas and Chandrasekaran [222] compared the runtime and accuracy of a classical SVM against two QSVMs, a kernel-based QSVM and a variational QSVM. The authors have determined that, for binary classification problems, a QSVM does not provide any substantial improvement over a classical SVM. However, in some multiclass classification cases the variational QSVMs exhibit a higher accuracy than classical SVMs. The QSVM multiclass classifiers are therefore promising as the number of available qubits increases. A similar conclusion has been reached by Zahorodko et al. [223] who expect that quantum-enhanced machine learning is suitable for classifying high-dimensional data sets in cases where especially binary classification can be processed by single-qubit systems in an efficient way.

5.3.8 Quantum Neural Networks

Artificial neural networks are machine learning models that mimic the way information is processed in biological brains. They are composed of small interconnected computational units called neurons which receive weighted input from other neurons. The weighted input received by a neuron is then typically summed up and subjected to a non-linear activation function. In other words, individual neurons can typically be thought of as a kind of binary linear classifier similar to the one in (34).

An important result due to Hornik [224] is that large enough neural networks are universal approximators. This mathematical statement is to be understood in the sense that they can approximate any Borel measurable function arbitrarily well. Their general universal approximation characteristic makes neural networks powerful tools for a wide range of artificial intelligence problems. Because of this, neural networks had their first boom period in the 1980s when the backpropagation algorithm for their training became available [225]. They then faded out of popularity because, back then, computers were not yet powerful enough to perform the extensive computations required in supervised neural network training. Over the past decade they returned to the limelight and are currently the most common tool in machine learning.

As the task of training large neural networks is computationally burdensome, quantum neural networks have become a popular topic among quantum computing researchers. Indeed, parameterized quantum circuits, i.e. quantum circuits whose gates or unitary operators come with tunable phase parameters, bear a certain resemblance to layered neural networks. Moreover, similar to the universal approximation theorem for neural networks, large or complex enough quantum circuits are known to be able to represent any target function with arbitrary precision [129]. Lin, Tegmark, and Rolnick also argue that data sets arising from measurements of physical systems will exhibit symmetry and locality so that there should exist less than exponentially large models that lead to useful results. Against this backdrop, variational quantum circuit models are intended to approximate solutions to a task at hand while also restricting the number of quantum operators and quantum circuit depth.

The proven universal approximation capability of classical neural networks crucially depends on the fact that their synaptic summations are subjected to non-linear activation functions. However, this poses a challenge for quantum neural networks. For instance, when data is encoded in the amplitudes of a quantum state, one cannot apply an arbitrary non-linear function without distorting it. Allcock et al. [226] therefore suggest algorithms for training and evaluating feed forward neural networks which are based on canonical classical feed forward and backpropagation procedures. Their algorithms rely on a quantum subroutine for approximating inner products between vectors which would yield training times quadratically faster in the size of the network than the classical counterparts. In addition, the authors assume that their approach is intrinsically resilient to overfitting because it quantum mimics classical regularization techniques. However, although the hybrid quantum-classical training procedure in [226] closely follows classical neural network training, it may not yet be practical. This is because Allcock et al. assume all sequential steps during their training procedure to be computed classically while quantum operations are only used for estimating the inner products in these steps. This requires their in- and outputs to be read from- and written to a QRAM which is impossible on current quantum computers.

Other works approach the aforementioned challenge with measurements. Beer et al. [227] propose a quantum analogue of a classical neuron from which they construct quantum feed forward neural networks capable of universal quantum computation. The basic idea is to consider a quantum perceptron to be an arbitrary unitary operator that maps m input qubits to n output qubits. Inputs are initialised in a possibly unknown mixed state and outputs in a fiducial product state, that is, in quantum state one can reliably reproduce with low variability. Using fidelity as a cost function, the authors propose a training procedure where perceptron unitaries are updated using phase shifts proportional to the loss incurred for a given pair of network in- and outputs. Regarding this procedure, they note that, if their quantum neurons are organized in layered architecture, updates can be accomplished layer by layer without having to access the full quantum circuit. They argue that this first of all reduces memory requirements and second of all decouples memory requirements from network depth. In other words, training efforts scale with the width of the proposed quantum neural network.

Running simulation experiments with quantum neural networks of moderate sizes, Beer et al. find their approach to generalize well to previously unseen testing data and also to be robust against noisy training data. A drawback with respect to present day quantum computers, however, is the assumption that quantum operators acting on arbitrarily many qubits were available. Given the present state of the art, this is technically not yet possible on existing devices.

Concerned with implementations on NISQ era quantum computers, Mitarai et al. [228] propose hybrid quantum-classical algorithm for quantum circuit learning. Their framework realizes task specific learning via iterative parameter tuning in circuits of a fixed depth. A theoretical analysis backed by simulation experiments reveals that their scheme can approximate nonlinear functions and the authors expect that the approach should be implementable on existing quantum computing devices.

Verdon et al. [229], too, focus on quantum neural network realizations for near-term quantum computers. In particular, they point out that it generally appears to be difficult to determine the initial parameters of a quantum circuit such that subsequent hybrid quantum-classical optimization will quickly converge towards local minima of the loss function under consideration. They therefore consider the use of classical neural networks which they train to generate initial parameters for the quantum learning process. They empirically find that this appears to allow for considerably faster variational quantum optimization in that the total number of optimization iterations required to reach a given accuracy is reduced considerably. The authors experimentally verify this for the quantum approximate optimization algorithm (QAOA) and for the variational quantum eigensolver (VQE) which they apply to problems such solving a graph max-cut task, the Sherrington-Kirkpatrick Ising model, or the Hubbard model. In fact, they observe that optimization strategies learned by the classical neural network seem to generalize across problem sizes. This leads them to expect that it may be possible to train their system on small problems that can be classically simulated and then transfer it to larger problems which are classically intractable. This way, the number of costly iterations in variational quantum-classical optimization for such settings could be reduced.

Indeed, the issue of how to initialize parameterized quantum circuits for quantum neural network training with hybrid quantum-classical gradient descent algorithms is critical. McClean et al. [230] observe that random initializations are popular and commonly considered by many researchers as they are simple as well as hardware efficient. However, they point to apparent inherent limitations of this approach when dealing with more than a few qubits. The authors demonstrate analytically as well as numerically that descent based training of even reasonably parameterized quantum circuits suffers from vanishing gradients and that there exist “barren plateaus in quantum neural network training landscapes”. This is to say that, during gradient based parameter optimization, expected values of observable required for the process tend to concentrate which causes their gradients to tend to zero. In other words, McClean et al. show that there is a high probability for gradient-based variational quantum neural network training to get stuck in local minima and thus to yield sub-optimal parameterizations. They argue that this is largely due to technical limitations of current quantum circuits and has to be taken into consideration when designing larger quantum neural networks.

Concerned with a practical implementation of parameterized quantum circuits on a trapped ion quantum computer, Zhu et al. [231] consider parameter optimization by means of particle swarm optimization and Bayesian optimization. Their work contains several noteworthy innovations. First of all, they consider the problem of training generative model circuits in the spirit of generative neural networks. These are neural networks which learn to produce new, plausible data points. Often, this happens in encoder-decoder architectures where training data are first compressed into low-dimensional representations and then decompressed into their original form. Known applications range from automatic text generation, over deep faked images or videos to the synthesis of plausible particle trajectories in high-energy physics simulations.

Zhu et al. argue that generative models can be expected to benefit considerably from potential quantum speedup and, second of all, describe a quantum circuit that is able to learn to reproduce patterns in the bars-and-stripes data set, a very simple, low-dimensional, yet common benchmark for experiments with novel techniques that is attributed to MacKay [20]. Tailored towards this data, they propose a quantum circuit design consisting of parameterized rotation and entangling operators which they adjust in a hybrid quantum-classical manner. In the classical parameter optimization steps, they work with particle swarm- and Bayesian optimization methods both of which are well established tools in classical machine learning. They implement their approach on a custom made quantum computing platform presented in [232] and observe that the convergence of their quantum circuit towards an optimized model critically depends on the optimization strategy. Their practical results suggest that it is practically possible to successfully train high-dimensional universal quantum circuits and that quantum neural networks may thus play a role in future generative modeling applications.

Leyton-Ortega et al. [233], too, are concerned with generative modeling for the canonical bars-and-stripes data and further investigate the observation that the performance of hybrid quantum-classical algorithms seems to depend on the choice of classical optimizer and on the circuit model. They argue that any conclusive (empirical) statement to resolve this issue requires to conduct extensive experiments with different optimization algorithms and circuit designs. To accomplish this in practice, they work with Rigetti's quantum cloud service and investigate the practical performance of data-driven quantum circuit training for different classical solvers and different circuit designs. For the latter, they consider different entangling qubit connectivity graphs and varying circuit depths. Their extensive experiments reveal that gradient-free optimization algorithms lead to much better results than gradient-based solvers. In particular, the former seem to be much less affected by the noise characteristics of existing quantum computing systems.

Another example of a real world implementation of quantum neurons can be found in work by Tacchino et al. [208]. They introduce a quantum neuron design where $m = 2^n$ dimensional data- and weight vectors are encoded using only n qubits which makes use of the exponential storage advantage of quantum computers. To generate entangled states they consider hyper-graph states [234] and prepare them using several controlled Z gates.

Non-linear outputs are realized by means of quantum measurements of ancilla qubits. The authors demonstrate the practical feasibility of their ideas through an implementation of the $n = 2$ case on an IBM Q5 quantum computer and find that their quantum neuron can successfully learn a simple bars-and-stripes pattern completion task.

Zhao et al. [168] observe that Bayesian methods such as Gaussian processes regression provide proven and successful machine learning models for reasoning under uncertainty and that they have recently been adapted to deep learning problems. They also observe that there exist proposals for quantum Gaussian process regression based on the HHL algorithm [235] and discuss how this allows for deep network training without backpropagation. Having established this connection, the authors propose a hybrid quantum-classical algorithm for Bayesian deep learning. They show how non-linear kernel matrices can be approximated in terms of polynomial series and then easily lend themselves to be used as quantum density operators. Using the HHL algorithm, whose very specific prerequisites are met by Gaussian process kernel matrices, their quantum subroutine achieves polynomially faster matrix inversion than classical methods and the authors simulate their algorithm on a Rigetti quantum virtual machine. For a considerably reduced setting with 2×2 Gaussian process kernel matrices, they also present implementations on the Rigetti 8Q-Agave and IBM Q5 quantum computers. Interestingly, they find that the probability of successful training is much higher on the IBM architecture which they attribute to its longer coherence times. In particular, the authors observe the probability of their protocol to train successfully to amount to 89% which is an encouraging result with respect to future efforts involving larger problem sizes.

Helmholtz machines are a class of neural networks that specifically allow for generative modelling [236]. They consist of two sub-networks, a bottom-up recognition network that maps input data to a distribution over hidden variables and a top-down generative network that generates novel data from an instantiation of the values of the hidden variables. The training of a Helmholtz machine usually happens in an unsupervised manner using an algorithm known as the “wake-sleep algorithm” [237].

Van Dam et al. [238] present a hybrid quantum-classical approach for Helmholtz machines. Their corresponding parameterized shallow quantum circuit models can be trained in a gradient-free manner using an optimization scheme based on the wake-sleep algorithm. The authors implement their system on the Quantum Inspire simulator and evaluate its practical performance on a reduced bars-and-stripes data set consisting of binary images of size 2×2 pixels. For this data they consider Helmholtz machines with four visible neurons and three hidden ones. They observe that their hybrid quantum-classical algorithm learns better network parameters than a corresponding purely classical implementation. They emphasize that their method can efficiently approximate the underlying probability distributions with only polynomially many evaluations of the quantum circuit and also expect their approach to be implementable on real quantum computing devices.

Boltzmann machines are yet another type of neural networks, in particular, a probabilistic extension of Hopfield networks. As such it is rather straightforward to conceive of quantum Boltzmann machines. This basically requires the definition of a suitable energy function whose minimizers constitute appropriate network parameters. Again due to their close connection to Hopfield networks, training can be realized via adiabatic quantum optimization [239] which, in turn, can be implemented on D-Wave computers [240].

Chen et al. [241] point out the internal probabilistic states of a Boltzmann machine can be translated into quantum states, which then leads to a purely quantum mechanical interpretation of Boltzmann machines called Born machines. These Born machines are yet another kind of generative model, and can be used to represent distributions of classical data in terms of quantum states in superposition. In this regard, Liu and Wang [242] remark that quantum sampling should be of lower computational complexity than sampling in corresponding classical implementations. They realize this by means of projective measurements of qubits and propose a gradient-based quantum-classical algorithm for optimization of a quantum circuit such that the likelihood of generated samples can be estimated. In simulation experiments with bars-and-stripes data, they observe their approach being able to learn data distributions, especially when deeper circuits are being used.

Coyle et al. [243] consider a restricted subset of Born machines, namely those whose Hamiltonians correspond to those encountered in Ising models. For these, they show that there is no efficient classical sampling scheme for the underlying quantum circuits and therefore propose a quantum-classical algorithm based on gradient descent. An innovative aspect of this approach is that the authors consider loss functions such as the Sinkhorn divergence or the Stein discrepancy. In experiments with practical implementation on Rigetti's forest platform (using a simulator as well as the Aspen quantum processing unit), the authors find that their procedure works well and reliably. As a potential future practical application of their generative modeling method, the authors point to the problem of quantum circuit compilation.

5.3.9 Federated and Distributed Quantum Machine Learning

In today's world, where cloud computing has established itself as a highly scaleable commodity tool for the processing of massive data sets, and where a great variety of data is available too, new concepts such as *federated learning* [244] promise further accelerated development. In federated learning, a central node holds the global model, it receives the trained parameters from client devices and then aggregates them to generate an updated and improved global model that is shared to all client nodes [244].

Chen and Yoo [244] deal with this concept in the context of quantum machine learning and discuss distributed training across several quantum computers. The authors expect a substantial reduction of training time and an advantage from the data security and data privacy perspective since the training is performed where the data is located. They demonstrate this federated training approach in experiments with hybrid quantum-classical machine learning models (which could potentially be translated to pure quantum machine learning models). In particular, in simulation experiments, they consider a quantum neural network combined with a classical pre-trained convolutional model on an image analysis problem. They observe that their federated training protocol does not sacrifice performance for accuracy (at least in the considered setting) and conclude that federated quantum machine learning might help to preserve privacy and to distribute computational efforts across arrays of NISQ devices.

Sheng and Zhou [245] introduced a protocol for *distributed secure quantum machine learning*. Their overall goal is to augment classical clients with basic added quantum computing technology to delegate remote quantum machine learning tasks to a quantum server in a manner such that privacy data is preserved. They present a distributed quantum machine learning protocol for the specific scenario where clients together with a remote server can assign two-dimensional data points to different clusters. Their protocol is secure in that it does not leak information relevant to this task and, similar to quantum information transmission, would immediately realize eavesdropping attempts. The authors suggest that this protocol could be extended to higher dimensional data and to bid data settings, but, for now, their work is rather conceptual and an interesting first step into a new direction.

5.3.10 Variational Quantum Algorithms

Above, we already saw several hybrid- or variational quantum computing approaches towards quantum machine learning. Examples included variational approaches towards quantum circuit design [143], quantum state preparation and loading [155], quantum linear system solving [175, 176], quantum support vector machine training [209], and ideas for the realization of quantum neural networks [210, 228, 229, 230, 231, 238]. Indeed, given the technical capabilities of present day NISQ devices, hybrid quantum-classical solutions, where quantum computers run model quantum circuits and classical computers repeatedly perform (statistical) optimization to adapt those models to a given task, currently appear to be the best strategy for quantum (assisted) machine learning in a manner that allows for harnessing quantum supremacy [246].

What all present such approaches have in common is that they involve parameterized quantum circuits. While these circuits may or may not have been specifically designed for a task at hand, they typically consist of tunable gates in sequence or in parallel. Most commonly, parameterized gates perform qubit rotations and the problem is to adjust their parameters such that the circuit as a whole performs the desired computation with high probability. In order to convey a better understanding of this general idea, we next discuss two prominent examples of variational quantum computing algorithms.

A well known and fundamental example of a variational quantum computing algorithm is the *variational quantum eigensolver* (VQE) introduced by Peruzzo et al. [85]. Motivated by the problem of having to estimate the ground state energy of certain chemical molecules, they consider the general problem of estimating the bottom eigenvalues of Hamiltonian operators. They observe that traditional quantum algorithms for eigenvalue computation relied on the phase estimation methods [115]. While these methods promise exponential speedup, they also generally require exponentially large numbers of quantum gates so that real world implementations on existing quantum computers are only possible for problems of small size [247].

Peruzzo et al. therefore propose a different approach and begin by observing the classical result that, for any Hermitian operator H, the Rayleigh quotient

$$r = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle} \tag{35}$$

is minimal exactly if $|\psi\rangle$ is the bottom eigenvector of H. Even more, if $|\psi\rangle$ is the bottom eigenvector of H, then it is known that r corresponds to the sought after minimum eigenvalue.

Based on this observation, the authors propose to consider a random, parameterized state $|\psi(\boldsymbol{\theta})\rangle$ and to systematically vary $\boldsymbol{\theta}$ to determine the minimizer of (35). In order to efficiently evaluate r on a quantum computer, they represent H as a linear combination of tensor products of Pauli operators and note that expected values of such operators can be estimated by means of local qubit measurements only. A considerable advantage their approach has over phase estimation techniques is that it requires only shallow quantum circuits as opposed to deep ones. It will therefore be less affected by short coherence times on existing quantum devices. A drawback is that the variational approach requires polynomially many executions of the circuits as opposed to just in the case of phase estimation approaches. For each of these calls, $|\psi(\boldsymbol{\theta})\rangle$ needs to be prepared and, in order to do this efficiently, the authors propose to let $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\phi\rangle$. Here, $|\phi\rangle$ is an appropriate, easily prepared reference state (for their molecular energy application, the authors consider the Hartree-Fock ground state) and $U(\boldsymbol{\theta}) = \exp(-iR(\boldsymbol{\theta}))$ where R is a rotation that acts on each qubit individually and can be decomposed into products of two simple Pauli operators. A quantum circuit which implements these computations can then be called in each iteration $t = 1, 2, \ldots, T$ of an outer loop executed on a classical computer which updates $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - d\boldsymbol{\theta}_t$ where, for instance, $d\boldsymbol{\theta}_t = r(\boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}) - r(\boldsymbol{\theta}_t - \Delta\boldsymbol{\theta})$ is an approximation of the gradient $\nabla r(\boldsymbol{\theta}_t)$ of the Rayleigh quotient r seen as a function of $\boldsymbol{\theta}$. Note, however, that gradient-free optimization schemes are possible as well. In fact, the latter do not seem to suffer from the phenomenon of “barren plataeus” [230] but often lead to better solutions than gradient-based schemes [143].

Peruzzo et al. demonstrate the practical feasibility of their method in experiments with an implementation on a custom made photonic quantum computer. In particular, they successfully show that ground state energies of the 4×4 Hamiltonian of a helium hydride ion can be calculated reliably.

The *quantum approximate optimization algorithm* (QAOA) due to Farhi et al. [84] is another variational method that was specifically developed to determine approximate solutions to combinatorial optimization problems. In the above discussion of adiabatic quantum computing, we already saw that potential solutions to such problems can often be encoded in terms of n binary variables z_j which have to meet m conditions $c_k(\boldsymbol{z}) \in \{0, 1\}$ and thus maximize $c(\boldsymbol{z}) = \sum_k c_k(\boldsymbol{z})$. Translating this setting into a formulation involving an n-qubit system $|s\rangle$ in a 2^n dimensional state space, Farhi et al. propose to consider an operator

$$U(C, \gamma) = \exp(-i\gamma C) = \prod_{k=1}^{m} \exp(-i\gamma C_k) \tag{36}$$

where C denotes the overall problem Hamiltonian. They also define an operator

$$U(B, \beta) = \exp(-i\beta B) = \prod_{j=1}^{n} \exp(-i\beta \sigma_j^x) \tag{37}$$

where $B = \sum_j \sigma_j^x$ and σ_j^x denotes the Pauli spin matrix σ^x acting on the jth qubit. The next crucial idea is to introduce a total of $2p$ angles $\gamma_1, \ldots, \gamma_p, \beta_1, \ldots, \beta_p$ and to consider the system

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(B, \beta_p)\, U(C, \gamma_p) \cdots U(B, \beta_1)\, U(C, \gamma_1)\, |s\rangle \tag{38}$$

which, as the authors emphasize, can be computed by a quantum circuit of depth $(m + 1) \cdot p$. The expected value of the Hamiltonian C under this “angle” state is $E_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta} | C | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle$ and the maximum value it can attain is called

$$M_p = \max_{\boldsymbol{\gamma}, \boldsymbol{\beta}} \; E_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \tag{39}$$

Crucially, the authors prove that, if $p \to \infty$, then $M_p \to \max_{\boldsymbol{z}} c(\boldsymbol{z})$. This then suggests a variational algorithm for approximately solving the given combinatorial problem: Choose p, initialize $\gamma_1, \ldots, \gamma_p, \beta_1, \ldots, \beta_p$, run a quantum circuit to compute $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ or to measure $E_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$, use this observation to classically adjust the current angle parameters, and repeat until $E_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$ does not improve anymore.

The reason why this idea works is that (38) can be seen as an approximation of an adiabatic quantum computing procedure with Hamiltonian $H(t) = (1 - t/T) \cdot B + t/T \cdot C$ [84].

Farhi et al. also consider several special cases where the problem Hamiltonian is of special structure and thus allows for particularly simple optimization routines. But, in general, the method is very versatile and can, in principle, be used to approximately solve any optimization problem that can be cast as a QUBO. The quality of the solution will depend on the choice of the parameter p. With respect to implementations on current NISQ devices, this allows for trading off circuit depth against approximation quality. Better approximations require larger values of p which lead to deeper quantum circuits which, in turn, may suffer from decohenrence. Smaller values of p entail shallower circuits which may be run in practice but also produce sub-optimal solutions.

Yet, with respect to quantum advantages that can be achieved by running QAOA, the jury is still out. According to a calculation by Dalzell et al. [248], QAOA would outperform any existing classical supercomputer for problems involving 420 qubits and 500 constraints. As practically relevant combinatorial optimization problems go, this is a rather moderate size, however, quantum computers that could work on problem sizes like this do not yet exist. Moreover, Akshay et al. [249] identified certain inherent limitations of the procedure. In particular, they found that its capabilities depend on the ratio of the number m of problem constraints and the number n of problem variables. In combinatorial optimization, it is known that problems where $\alpha = m/n < 1$ tend to have several to many solutions whereas problems with $\alpha = m/n > 1$ have only few solutions if at all. For the latter kind of problems, Akshay et al. found that, even for large p, QAOA does not seem to work well. The authors refer to this as a (solution) reachability deficiency of QAOA and concede that further research is required to better understand the phenomenon.

Overall, however, variational- or hybrid quantum-classical approaches can considerably reduce quantum computing resources (circuit depth, coherence time, qubit counts) required to run quantum machine learning methods in a stable manner. Importantly, while the state spaces considered in this context are exponentially large, parameterized circuits typically have fixed structures (mainly laid out by human experts) and the number of their parameters only scales polynomially with the number of qubits. In fact, a common current design principle for parameterized circuits is to consider NISQ hardware efficient layouts, i.e. circuits of low qubit connectivity and with comparatively simple gates. Here, tensor network designs are of increasing interest, but another strategy is to apply variational algorithms to design simple circuits on which to run variational algorithms.

The appeal of variational quantum algorithms of parameterized quantum circuits to those who develop quantum neural networks can likely be attributed to the fact that both, deep neural networks and parameterized quantum circuits, consist of basic computational units arranged in layered structures. Moreover, both involve the use of classical optimization algorithms to adjust their parameters to a given, specific problem. Some authors therefore even refer to parameterized quantum circuits as quantum neural networks per se. However, there are differences which make this terminology questionable.

First of all, the basic computational units in a quantum circuit are unitary operators rather than non-linear functions as in the case of neural networks. Above, we already discussed that the capabilities of neural networks crucially depend on the fact that the activation functions of neurons are non-linear. Non-linear computations in a quantum circuit can be realized in a manner that preserves coherence, e.g. through entanglement, or in a manner that destroys coherence, e.g. through qubit measurements. In particular in combination with ancialla qubits, these operations allow for modelling the behavior of artificial neurons.

Second of all, internal states of a quantum circuit cannot be read out in a manner that would preserve their quantum nature. Contrary to neural networks, it is therefore impossible to compute local gradients within a quantum circuit. Consequently, it is impossible to apply methods like the backpropagation algorithm to train quantum circuits. As of this writing, there thus do not exist any workable proposals for how to adjust the parameters of a parameterized circuit without truly external supervision through an independent (classical) optimization algorithm.

5.4 Quantum Machine Learning for Quantum Data

If the prospects of quantum enhanced machine learning for classical data are most exciting for machine learning researchers, then the prospects of quantum machine learning and quantum analytics for quantum data are likely most exciting for physicists and chemists. This is because the simulation of physical or chemical systems or processes on quantum computers is widely seen as another promising application of quantum computing [250]. Such quantum computing simulations produce quantum data whose subsequent analysis would either require to measure and read them into classical computers or, preferably, could itself involve quantum algorithms. Quantum machine learning for quantum data thus refers to the idea of using quantum machine learning methods to process quantum mechanical data on quantum devices.

For example, Grant et al. [210] note that their hierarchical quantum circuits for binary classification can be applied to the problem of quantum state classification. Basheer et al. [199], too, point out that their quantum circuits for k-nearest neighbor classification can be directly used on quantum data. Chen et al. [251] particularly focus on the question of how to classify quantum data and consider the specific problem of distinguishing between pure and mixed states. To accomplish this, they train a parameterized quantum circuit and report that their system performs close to optimal on the training data and, more importantly, also generalizes well to previously unseen inputs. Since they are dealing with the classification of genuine quantum mechanical data, they conclude that their approach is among the first to successfully solve a learning problem for which there is no classical counterpart.

The term quantum state tomography refers to the general problem of predicting (probabilities of) outcomes of measurements of hidden quantum states. Traditional approaches to this estimation problem would require exponentially many measurements but Aaronson [252] observes that the problem could also be thought of a statistical learning problem. He then proves that estimates can be obtained from only linearly many measurements. In the meanwhile, together with a team of coworkers, he verified this theoretical expectation in practice [253]. Working with a custom made photonic quantum computer, they experimentally demonstrate this linear scaling behavior for their system in optical systems and conclude that computational learning theory provides a useful tool for the analysis of information that is of genuinely quantum mechanical nature.

Legeza and Solyom [254] investigate quantum data compression for finite quantum systems for cases with dependent density matrices. The authors demonstrated a connection between the entropy of the left or right block and dimension of the Hilbert space of that block.

Romero et al. [255] are concerned with the problem of autoencoding quantum data. They propose a simple parameterized quantum circuit that can act as an autoencoder neural network, i.e. can compress and decompress its input data. In simulation experiments, they train their model in a quantum variational manner so that it learns to compress a set of quantum state data for which classical compression algorithms supposedly do not work. In particular, they consider the problem of compressing Fermionic wave functions, i.e. eigenstates of a number operator, of a system of n quantum particles, compare their results to analytically computed predictions, and find it to work well. Ding et al. [256], too, are concerned with autoencoding quantum states and propose a corresponding quantum circuit composed of quantum adders. In an experimental implementation in the Rigetti cloud, they consider states of three superposed qubits and find their method to produce results that agree well with theoretical expectations.

In classical machine learning, autoencoders are often used as generative models and have been extended towards more capable architectures such as generative adversarial networks (GANs). In this spirit, Benedetti et al. [257] propose a system composed of two parameterized quantum circuits for generative modeling. They then use their system to sample unknown pure quantum states. Numerical simulation experiments suggest that the approach works well and the authors foresee applications in quantum state tomography. Similarly, Hu et al. [258] demonstrate that quantum state synthesis based on generative adversarial learning is practically possible. They implement their approach on a superconducting quantum circuit and find that it can be trained to generate high fidelity quantum output from quantum input

In their article “quantum data processing and error correction”, Schumacher and Nielsen [259] deal with the behavior of noisy quantum information channels. The authors introduce the concept of coherent information which specifies a (per quantum information processing non-increasing) quantity measuring the amount of quantum information conveyed in the noisy channel. This quantity indicates the necessary and sufficient condition for the existence of perfect quantum error correction. Quantum error correction, in turn, is of importance in the NISQ era since existing devices suffer from noise, unreliable operations, and decoherence so that methods for generating fault-tolerant result are sought after.

6 Characteristics of Quantum Machine Learning Methods

The ever faster growing literature on quantum machine learning and the many success stories reported therein seem to suggest that quantum computing for computational intelligence is just about to break into the mainstream. However, as of this writing, the content of many reports, including many of those surveyed above, has still to be taken with a grain of salt. The two major caveats are that reported results might be exaggerated or that reported results abstract away from technical reality and assume the existence of universal quantum computers whose hypothetical capabilities far exceed what is possible with current NISQ era devices.

As a testimony to the former, we refer to Aaronson's article "Read the Fine Print" [260] in which he critiques what he calls the "quantum machine learning mini-revolution". In particular, he presents a critical analysis of what the HHL algorithm [119] for linear system solving is capable of and contrasts that with how it is commonly perceived by enthusiasts in the public and the scientific community.

Aaronson observes that HHL is often seen as a quantum algorithm for solving general linear systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ in a time exponentially faster than classically possible. That is, if $\boldsymbol{A} \in \mathbb{C}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{C}^n$, then HHL is often seen as a method for finding $\boldsymbol{x} \in \mathbb{C}^n$ in a time proportional to $\log n$ rather than to n^2. He then points to the fine print and emphasizes that HHL is actually a method for creating a quantum state $|x\rangle$ which approximately represents the sought after solution $\boldsymbol{x}$. He further emphasizes that the quantum speedup of the method hinges on the assumption that a state $|b\rangle$ representing $\boldsymbol{b}$ can be prepared efficiently. This *encoding problem* may be easy in certain special cases, for example, if $\boldsymbol{b}$ is a unit vector whose entries have about the same magnitude. In general, however, state encoding will require efforts at least polynomial in n. Moreover, once $|b\rangle$ has been prepared, it must be coherently subjected to the operator $\exp(-iAt)$ for a period of time proportional to $ks/\epsilon \cdot \log n$. Here, k and s measure condition number and sparseness of matrix $\boldsymbol{A}$ and ϵ indicates the approximation error users are willing to tolerate. Aaronson finally stresses that not every matrix is well conditioned and sparse enough to meet the implicit requirements under which the HHL algorithm is guaranteed to run efficiently.

Indeed, the general utility of HHL has been scrutinized before. For instance, Childs [261] observes that producing a quantum state $|x\rangle = \boldsymbol{A}^{-1}|b\rangle$ is not tantamount to solving the original linear system. Rather, he points out the *decoding problem* and notes that obtaining $\boldsymbol{x}$ from $|x\rangle$ requires $\mathcal{O}(n)$ quantum measurements. This leads him to conclude that, when it comes to really solving a system of linear equations, present quantum algorithms do not yet have an exponential edge over classical approaches.

Both Aaronson and Childs note that the inventors of the HHL algorithm acknowledge all of this. Indeed, the above shortcomings and restrictions might be the reason why Harrow, Hassidim, and Lloyd [119] remark that there are situations in which we may not be interested in the actual solution $\boldsymbol{x}$ but only in some summary statistic of it, such as, say, the sum of its entries. They also stress that, in situations like these, quantum algorithms may work much faster than their classical counterparts and move on to argue that any possible quantum computation could be understood as an instance of linear equation solving. This argument, in turn, would either imply that classical computers should be able to efficiently simulate quantum computations (which they are not) or that quantum computers have an inherent linear equation solving advantage over classical computers. As the latter has not yet been conclusively demonstrated in general, Childs is critical about hardness results such as this one, and questions their role as a means of arguing for quantum supremacy in applied settings.

Overall, Aaronson concedes that the HHL algorithm and its decedents constitute theoretical as well as practical progress in quantum computing. At the same time, he emphasizes the need to carefully examine conditions or assumptions required for a quantum algorithm to achieve exponential speedup. Especially if effort required for in- and output or pre- and post-processing of data are factored in, many quantum computing solutions turn out to be only polynomially faster than classical algorithms. As Tang's former supervisor, Aaronson certainly knows what he is talking about. We therefore recall that Tang famously found a novel classical algorithm which—after appropriate pre-processing steps—works only polynomially slower than its quantum counterpart. From this, she concluded that state preparation assumptions in a quantum computing solution need to be matched against classical pre-processing assumptions before any claims can be made about significant quantum speedup.

To add further credence to this conclusion, we mention another example from our own experience. Recall that reports such as [209] or [226] describe classifiers which involve quantum subroutines for computing approximate inner products between quantum state vectors. However, if "approximate" inner products between high dimensional vectors are all that is asked for, we note that there exist classical methods which—after appropriate pre-processing steps—can compute them in $\mathcal{O}(1)$, i.e. in a time independent of the data dimensionality [262]. Especially in cases where inner products need to be evaluated repeatedly, e.g. in nearest neighbor searches, the costs for pre-processing quickly amortize and potential quantum advantages may become less significant. Granted, techniques such as in [262] appear not to be widely known but they exist and should be taken into account by quantum machine learning researchers.

The second caveat we mentioned in the introductory paragraph has to do with the common assumption that universal quantum computers are taken as a given when developing quantum computing algorithms. Yet, given the current state of technical developments, this is a far reaching assumption as many of the expected capabilities of universal quantum computers can not be realized by current NISQ era devices. Hence, to assess the actual technical feasibility of proposed quantum machine learning approaches, we have to check them against a catalogue of criteria. Important such criteria will be discussed in the following.

6.1 Hardware Requirements

When quantum algorithms are designed, properties of the hardware at hand are usually not taken into account. This is not surprising, since abstracting the actual hardware away is at the core of computer science. Fundamental results about complexity, data structures, and algorithms rely on theoretical models of computation, e.g., Turing machines and other types of automata. This is clearly sufficient when the correctness of an algorithm, its asymptotic runtime, or bounds on its memory consumption are being analyzed. In these scenarios, it is not necessary to bother about the clock rate of a processor, the amount of available main memory, or the data width of some bus architecture. In fact, theoretical machines possess an infinite amount of memory. Nevertheless, when algorithms have to run on real computers, all these constraints materialize and have to be considered for implementation. Algorithms can be equivalent under a theoretical model while their actual number of consumed clock cycles on a specific processor can differ by a large amount. Hence, one of the algorithms would consume far more energy and they stop being equivalent in appliances where a steady energy supply cannot be guaranteed, e.g., in the context of autonomous systems. Moreover, without specific tuning of an algorithm, state of the art GPUs cannot unfold their potential and many of their computational capabilities will lie idle [263]. Some of these practical considerations have found their way back into the theoretical community. One striking example are cache-oblivious algorithms [264], where the existence of a memory hierarchy is introduced to the theoretical analysis. Moreover, the explicit consideration of hardware properties also found its way to machine learning, e.g., machine learning models which are designed to rely only on integer arithmetic [265].

While the explicit incorporation of computational architectures has led to various successes in classical computer science, similar ideas are rather unknown in the quantum machine learning literature: When QML methods are derived, there is no treatment of particularities of the underlying quantum compute architecture. Authors do neither discuss whether, e.g., the usage of a Toffoli gate in a quantum circuit is problematic, nor is it discussed if using $\mathcal{O}(\log n)$ ancilla qubits leads to a practical relevant method. Especially because practical quantum computing hardware is still in its infancy and resources are scarce, real world constraints must be considered in order to understand if a specific QML algorithm is viable.

6.1.1 Quantum Memory

Some low-end classical processors like microcontrollers are not equipped with a floating point unit or units for processing wide single instruction multiple data vector operations. When an algorithm designer assumes the availability of such functional units, the algorithms are unlikely to run satisfactorily on other processors. The same happens in the design of quantum algorithms: Various QML methods require the availability of some sort of quantum memory. Given the current technical state of the art, the existence of an exact quantum memory is at least questionable due to the no-cloning theorem [88]. The theorem asserts that there is no unitary U for which two non-orthogonal quantum states $|\psi\rangle$ and $|\phi\rangle$ exist, such that

$$U(|\psi\rangle \otimes |x\rangle) = |\psi\rangle \otimes |\psi\rangle \qquad \text{and} \qquad U(|\phi\rangle \otimes |x\rangle) = |\phi\rangle \otimes |\phi\rangle$$

whereas $|x\rangle$ is some arbitrary state that we want to overwrite with the content from $|\psi\rangle$ or $|\phi\rangle$. In short, we will not be able to construct a unitary operator that allows us to copy the content of arbitrary qubit registers $|\psi\rangle$ and $|\phi\rangle$ to a target register $|x\rangle$. It is important to understand that the no-cloning theorem prohibits the existence of a quantum computing system in which $|x\rangle$ is a working register and $|\phi\rangle$ and $|\psi\rangle$ are states which are stored in some type of quantum memory—making it effectively impossible to load data from the memory or to write data into the memory.

Nevertheless, a whole line of QML research relies on this concept, including quantum recommendation systems [112], quantum deep convolutional neural networks [266], quantum gradient descent, methods for solving linear systems, and methods for ordinary least squares regression [267]. The primary building block of these approaches is the QRAM [268]. While being frequently cited, fundamental issues like the no-cloning theorem are not discussed in the paper that introduces QRAM. Instead, the authors refer to "quantum memory elements" without describing how they can be implemented. Moreover, a three-level quantum system is required to realize the routing in the so-called bucket-brigade memory architecture. While this is not a problem per se, commercial quantum computers with hardware for ternary quantum states are not available. Hence, rendering algorithms which rely on these concepts as not viable. Finally, 2^n memory slots are assumed to be available, where n is the number of bits in the address register. Clearly, in the light of currently available hardware and under consideration of vendor roadmaps, the availability of a large number of qubits is not given at the time of writing.

Physical implementations of superconducting qubits reside on the chip at fixed locations and are connected via a well-defined pattern, the so-called *connectivity structure*. Any specific structure originates mainly from practical considerations. In IBM Q Systems, the predominant connectivity is known as *Heavy Hexagon* structure, shown in Fig. 4 (a). These low-degree graphs are designed to minimize the possibility of frequency collisions and optimize the hardware performance within superconducting qubit architectures. The larger the number of neighbors of a qubit, the more frequencies are required to realize two qubit gates using cross-resonance interaction. A specialized family of error correcting codes for this structure exhibits outstanding properties with respect to decoding and frequency collisions [269].

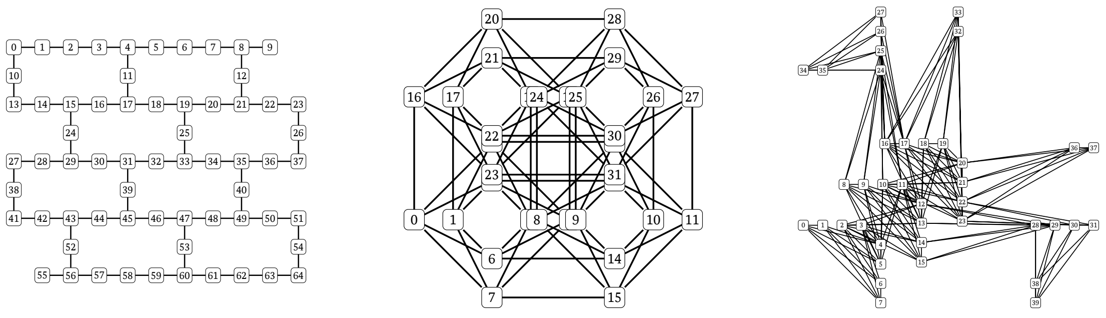
Beside its favorable properties for error correction, the connectivity of a quantum gate processor has direct impact on the depth of actual quantum circuits. To see this, one has to consider the transpilation of user specified circuits. During transpilation an input circuit is compiled to a sequence of native gates such that all operations agree with the connectivity structure and noise properties of a specific quantum processor. For any non-trivial circuit, a series of transformations must be conducted to make it compatible with a given target device, and optimize them to reduce the effects of noise on the resulting outcomes. Due to high-dimensional noise distributions and limited gate sets, rewriting quantum circuits to match hardware constraints and optimizing for performance can be far from trivial. Quantum circuit compilation can have a non-linear control flow and exhibits complex branching patterns. Most importantly, it encompasses the decomposition of gates involving three or more qubits into 2-qubit gates. Clearly, the heavy hexagon structure is pairwise and hence contains no connection between three or more qubits. As a direct result, an apparently "shallow" quantum gate circuit, consisting of a single unitary operation among 10 qubits, can thus eventually exhibit a high depth. On the other hand, a high depth requires long decoherence and dissipation times of the system, and thus, might not be viable.

6.1.2 Qubit Connectivity

To understand the impact on QML methods, let us consider quantum kernel machines from Sec. 5.3.7. The power of the quantum kernel comes from the fact that data is mapped into a feature space whose dimension is exponential in the number of data dimensions. It is important to understand that the dimensions of the feature map shall not be independent of each other. In other words, all qubits that underlie the feature space representation shall be entangled with each other. For quantum kernel machines [218], this is realized by considering the following unitary:

$$U_{\phi(\boldsymbol{x})} = \exp\bigg(-i \sum_{S\subset[n]} \phi_s(\boldsymbol{x}) \prod_{v\in S} \sigma_z^v\bigg)$$

Here, $\phi_s(\boldsymbol{x})$ denotes a data dependent angle. The expression $S \subset [n]$ denotes all possible subsets of n qubits. We hence conjecture that $U_{\phi(\boldsymbol{x})}$ creates an entanglement between all qubits. However, we know that the process of transpilation will break this down into 2-qubit gates, and thus, a very deep and most likely not viable circuit. The authors of [218]



(a) Heavy hexagon, 65-qubit. **(b)** Chimera (C2), 32-qubit. **(c)** Pegasus (P2), 40-qubit.

Figure 4: Qubit connectivity overview.

are, of course, aware of this particularity and propose to restrict the subsets S to those of size two, hence, taking the actual hardware structure into account. While this solution is perfectly reasonable, it greatly limits the expressiveness of the resulting kernel, and thus the accuracy of any QML method that relies on this kernel construction.

In case of the D-Wave quantum annealer, the *Chimera* and *Pegasus* structures define the qubit neighborhoods. They arise from considerations about a high inter-qubit connectivity. Formal definitions in terms of edge generating functions of the graphs shown in Figs. 4 (b) and (c) together with a brief discussion of their graph theoretic properties are provided in [270]. While connectivity seems to be a rather technical property of a quantum annealer, it has an immediate impact on the problems that can actually be solved on that chip. More precisely, the structure imposes constraints on the sparsity pattern of the QUBO matrix that encodes the problem that has to be solved. In Chimera, the qubits can couple to up to $\delta = 6$ other qubits. In Pegasus, each qubits can couple to up to $\delta = 15$ other qubits. This implies that each row of the QUBO matrix may only contain δ non-zero entries. Moreover, those entries must reside at fixed positions within each row. Revisiting the insights from the previous section, many QML methods that are based on quantum annealing require a completely dense QUBO matrix, e.g. the clustering method presented in [73]. There, the dimension of the QUBO problem is equal to the number of data points that have to be clustered and the QUBO matrix itself stems from the kernel matrix between all pairs of data points, and hence is dense.

Limited connectivity can be mitigated by introducing so-called chains into the problem's Hamiltonian. That is, the optimization problem is rephrased such that multiple qubits are enforced to take on the same value in the system's ground state, thus effectively increasing the fan-out of that qubit while reducing the total number of available qubits. To understand the consequences, consider the D-Wave Advantage 4.1 Quantum Annealer with 5627 qubits. When a complete dense QUBO matrix is required, chains of 17 qubits are introduced which results in an effective dimension of 177. The maximum dimension of the problem is hence reduced by orders of magnitude, severely reducing the number of data points that can be processed by the aforementioned kernel clustering approach.

6.1.3 Qubit Count

The number of qubits in real world systems is ever increasing. At the time of writing, at most 127 qubits are available for quantum gate devices, e.g., the IBM Eagle processor, and at most 5627 qubits for quantum annealing, e.g., D-Wave Advantage 4.1.

As mentioned in the previous section, the problem structure has a direct impact on the number of qubits that are actually available to a quantum annealer. When the problem dimension exceeds the number of available qubits, the problem is too large to be embedded in its entirety on the quantum processor. Instead, the problem is sequentially decomposed into smaller problems, each of which can fit on the quantum processor, with variables selected by highest impact on the objective function. Clearly, variables cannot be selected solely by maximal impact on the objective function, since those qubits may not be connected at all. In order to represent a local structure of the problem, traversal techniques such as breadth-first (BFS) or priority-first selection (PFS) can capture features that represent local structures within a problem. Instead of selecting the working set of qubits via their impact, *cutset conditioning* can be applied. That is, the values of a subset B of variables (the "cutset") are fixed, effectively partitioning the set of variables $[n] = A \cup B$ with $A \cap B = \emptyset$. Thus, the QUBO problem $\text{argmin}_{\boldsymbol{s}\in\{-1,+1\}^n} Q(\boldsymbol{s}) = \boldsymbol{s}^\intercal \boldsymbol{Q}\, \boldsymbol{s} + \boldsymbol{q}^\intercal \boldsymbol{s}$ becomes

$$\underset{\boldsymbol{s}_A\in\{-1,+1\}^n}{\text{argmin}} \quad Q(\boldsymbol{s}_A, \boldsymbol{s}_B)$$

Finding the optimal setting for $\boldsymbol{s}_A$ while values in $\boldsymbol{s}_B$ are fixed breaks the problem into separate components that can be solved independently. When A is chosen carefully, each independent problem is small enough to be solved on the quantum annealer. This procedure has to be repeated multiple times, choosing different subsets A. The general procedure is outlined in [271].

QUBO problems can indeed be solved on quantum gate computers, too. Based on the Variational Quantum Eigensolver [85], the Quantum Approximate Optimization Algorithm [84] provides specialized variational circuits, designed for addressing combinatorial optimization problems. The underlying variational forms have later been improved, yielding the Quantum Alternating Operator Ansatz [272]. As opposed to the quantum annealers with fixed connectivity, QAOAs dimension does not degrade as a function of the QUBO density. However, the number of qubits in contemporary quantum gate systems is lower than in quantum annealers. Thus, the same splitting techniques can be applied. Hence, larger optimization problems can be addressed even in case of a relatively low number of qubits.

For non-iterative or non-variational algorithms, it is in general not possible to split any quantum gate computation into smaller parts—mostly because entanglement cannot be maintained among the sub-problems. A reasonable direction can be the minimization of required qubits. However, the number of required "data" qubits or "input" qubits is problem specific. Methods for embedding high-dimensional points into a lower-dimensional manifold are available aplenty, but due to their highly approximate nature, it is unclear if the resulting embedded points still contain enough information. Assuming that the data qubits are fixed, the other major source of qubit utilization are so-called *ancilla qubits*. In a quantum computation, there is no way to deterministically construct a prescribed state unless one is given access to qubits whose original state is known in advance.

As an example of this concept, consider a matrix M whose singular values are upper bounded by 1. Let us denote the singular value decomposition of M by RDV. Then, the matrix

$$U = \begin{bmatrix} M & R\sqrt{I-D^2}V \\ R\sqrt{I-D^2}V & -M \end{bmatrix}$$

is unitary. This can be verified by observing that $UU^\dagger = I$. Note, however, that the dimension of the matrix has been doubled. Interpreting M as an operator that shall be applied on some state $|\psi\rangle$, we have to extend the state by one additional qubit $|0\rangle_a$, such that $U(|\psi\rangle \otimes |0\rangle_a)$ applies M to $|\psi\rangle$. A generalization of this idea is the linear combination of unitaries [273]. Unitaries are not closed under summation. However, a sum of m unitary matrices can be extended to an unitary operator by using $\log m$ ancilla qubits.

Ancilla qubits appear frequently in quantum machine learning [113, 114, 234]. However, care must be taken when the number of ancillas is provided in an asymptotic manner, e.g., [117, 274]. Given n input qubits, $\log n$ ancilla qubits can be reasonable. Nevertheless, $\mathcal{O}(\log n)$ may contain large constants, such that the number of required ancillas actually exceeds the number of available qubits by a large margin.

6.1.4 Quantum Circuit Depth

With the currently available NISQ technology, the result of a quantum computation on a physical device may deviate significantly from the expected outcome. A fundamental reason is that the quantum computer, despite all technical efforts, is not perfectly isolated and interacts (weakly) with its environment. In particular, there are two major effects of the environment that can contribute to computational errors: dissipation and decoherence in the sense of dephasing [275, 276]. Dissipation describes the decay of qubit states of higher energy due to an energy exchange with the environment. Decoherence, on the other hand, represents a loss of quantum superpositions as a consequence of environmental interactions. Typically, decoherence is more dominating than dissipation. In current quantum gate devices, these effects are captured by the quantities $T1$ and $T2$. Both can be interpreted as decay constants. That is, the probability that a qubit will stay in state $|1\rangle$ after time t is proportional to $\exp(-t/T1)$. Similarly, $T2$ indicates the decay constant for which an initial state $|+\rangle$ will evolve into an equal classical probabilistic mixture of the $|+\rangle$ and $|-\rangle$ states, so that one can no longer confidently predict the state. That is, it's the autocorrelation time after which the initial and final states become uncorrelated.

Decoherence effectively limits the length of the longest computation and thus the allowed depth of a quantum circuit. In other words, the more operations have to be performed on a qubit register, i.e. the deeper a quantum circuit has to be, the likelier it is that decoherence will happen. As explained in Section 6.1.2, transpilation will replace any unitary operator which acts on 3 or more qubits into a series of one-qubit and two-qubit gates. This implies that a single high-order operator might result in a circuit that is actually too deep to be executed by a system. Despite decoherence and dephasing constants $T1$ and $T2$ being available for most quantum processors, they are not utilized to adapt quantum machine learning methods to the underlying hardware and to account for the fact that the maximum computation time is limited.

Quantum devices based on adiabatic evolution are affected by decoherence as well [277]. The direct effect is a shorter usable annealing time and thus the time until a solution to the user specified optimization problem must be delivered. To see why this is significant, one has to recall that simple optimization problems can already be solved by classical computers. Thus, we are mostly interested in quantum solvers for hard optimization problems. However, hard problems, e.g., measured via the Hamiltonian's spectral gap, require a longer computation time—which is prohibited by the upper bound on the annealing time.

6.2 Algorithmic Requirements

We have discussed a series of pitfalls that arise in the design of quantum machine learning algorithms due to particularities of the quantum hardware. Of course, issues can also manifest solely from algorithmic aspects. In classical computing, efficient data structures and data types with a sufficient precision must be used. Components like pseudo random number generators and algebra sub-routines must be reliable and numerically stable. Parallel code has to be checked for race conditions and other problems that might emerge due to concurrency.

In quantum machine learning, similar issues can show up. In what follows, we discuss pre- and post-processing of inputs and outputs, stochasticity of the quantum computation, and noise sensitivity.

6.2.1 Data Pre- and Post-Processing

The question of how to encode data for training or inference with quantum machine learning methods so as to benefit from quantum advantages still awaits a final answer—if possible at all. Expected advantages, i.e. quantum speedup, may not emerge in practice if data pre-processing would require exponential efforts.

The root of the pre-processing problem lies in the preparation of states. To see this, consider the n-qubit state

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_j}|j\rangle$$

where $|j\rangle$ denotes the computational basis state that corresponds to the binary encoding of the integer j, and p_i is an arbitrary but fixed, classical probability mass function over $\{0,1\}^n$. Clearly, various types of data can be encoded into that representation: e.g., p_j could represent

1. (normalized) intensity values of pixels in an $2^{n/2} \times 2^{n/2}$ image,
2. term frequencies over some text corpus with a vocabulary of size 2^n,

3. or the probability of observing a traffic jam on a specific set of street segments.

Related encodings indeed appear in the literature [278, 279, 280]. In that representation, a greyscale image in 4K resolution (3840 x 2160 pixel) would occupy $\lceil \log_2(3840 \times 2160) \rceil = 23$ qubits. It is, however, important to understand that preparing the state $|\psi\rangle$ requires resources that are proportional to the dimension of the Hilbert space of the underlying qubit register. In the above example $n = 23$ was small enough such that processing 2^n values is feasible. If n is large, we cannot simply prepare $|\psi\rangle$. Even if a quantum algorithm would yield an exponential improvement over the best known classical algorithm given input state $|\psi\rangle$, it would not be possible to gain any advantage due to the complexity of preparing the initial state.

By assuming efficient integrability of an input probability distribution p_i, Grover and Rudolph explain how to efficiently generate the quantum state in a coherent fashion such that subsequent processing can be conducted on a quantum device [154]. However, their assumptions are not satisfied in most setups.

Recently, a classical model of computation was introduced which assumes that we can efficiently sample instances from a data set, where the probability to draw a specific data point is proportional to its ℓ_2-norm. It can be shown that this is a natural analog to quantum algorithms that assume efficient state preparation of classical data. Based on this reasoning, classical versions of quantum algorithms for principal component analysis [281] and nearest-centroid clustering [198] can be devised. The runtimes of these classical algorithms are only polynomially slower which suggest that the exponential speedups of their quantum counterparts are an artifact of state preparation assumptions [282].

The same issues indeed arise when we want to read the result of a quantum computation. When the full Hilbert space representation of the state encodes our desired outcome, an exponential number of results has to be measured. Thus, even if the quantum computation itself delivers an exponential speedup, this advantage will vanish when all probability amplitudes of the final state have to be estimated.

6.2.2 Statistical Error

Statistical learning theory (SLT) is an integral part of machine learning. It may be the core distinguishing property between machine learning and plain numerical optimization. The theory itself is universal and independent of the underlying compute architecture. Through the eyes of statistical learning theory, there is nothing special about quantum machine learning. QML has to obey the laws and limitations predicted by SLT. At a first glance, this does not sound harmful. Nevertheless, some QML algorithms claim speed-ups over classical methods which can easily be falsified by basic insights from SLT. In what follows, we explain the basic reasoning behind this.

Assume that we try to find a function f such that the expected loss ℓ, also known as *risk*, is minimized:

$$R(f) = \mathbb{E}_{\mathbb{P}}[\ell(Y, f(X))]$$

Here, X is a random variable that represents the observed data or features, and Y is a random variable representing the class label (or regression target). The pairs (X, Y) follow the distribution $\mathbb{P}$. In practice, $\mathbb{P}$ is indeed unknown, but a data set $\mathcal{D}$ that contains samples from $\mathbb{P}$ is available—allowing for the computation of the *empirical risk*:

$$\tilde{R}(f) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(y, f(x))$$

Let now $\mathcal{H}$ denote a machine learning method, e.g., $\mathcal{H}$ may contain all decision trees, all support vector machines with the same fixed kernel, or all neural networks with the same fixed network structure. Under rather mild assumptions, $\mathcal{H}$ contains a minimizer of $\tilde{R}$ [283]. Let us denote the empirical risk minimizer by $\tilde{f} \ = \ \arg\min_{f\in\mathcal{H}} \tilde{R}(f)$. The generalization error (also known as estimation error) can be expressed as distance between both minimizers in terms of R:

$$R(\tilde{f}) - \inf_{f\in\mathcal{H}} R(f) \leq 2 \sup_{f\in\mathcal{H}} \ \tilde{R}(f) - R(f) \ \leq \Theta\left(\sqrt{\frac{c(\mathcal{H}) - \log\delta}{|\mathcal{D}|}}\right)$$

where Ω denotes asymptotic equivalence and $\delta \in (0, 1)$ is a probability. The last inequality holds with probability $1 - \delta$ over the choice of the data set. Moreover, $(\mathcal{H})$ denotes a measure of complexity or expressivness of $\mathcal{H}$ (e.g., VC dimension or Rademacher complexity [283]). When $\mathcal{H}$ and δ are treated as fixed constants, the estimation error is upper bounded by $\Theta(\sqrt{1/|\mathcal{D}|})$. Hence, in order make use of all available data, any additional error that is introduced by a QML method shall not be larger than $\mathcal{O}(\sqrt{1/|\mathcal{D}|})$—no matter if it is generated by an approximate optimization, measurement noise, or any other type of algorithmic error. When we denote the learned function that incurs an additional algorithmic error by $\tilde{f}_\varepsilon$, we have

$$R(\tilde{f}_\varepsilon) - R(\tilde{f}) = \Theta(\sqrt{1/|\mathcal{D}|}) + \underbrace{(\tilde{R}(\tilde{f}_\varepsilon) - \tilde{R}(\tilde{f}))}_{\text{Algorithmic error}} .$$

While this insight might not sound significant, it can be used to prove that quantum learning algorithms must have at least polynomial runtime in the dimension of the training data, and therefore cannot achieve exponential speedups over classical polynomial time machine learning algorithms—even when special data storage like QRAM is assumed [284].

To understand the impact, consider quantum least squares (QLS) regression [267]. The QLS algorithm is approximate. Its output is a state $|\hat{w}\rangle$ while the state that contains the true regression weights are $|w\rangle$. For the error, we have $||\hat{w}\rangle - |w\rangle| \leq \gamma$. According to [179], the runtime is $\mathcal{O}(\kappa^c\gamma^{-\beta}$ polylog$(|\mathcal{D}|))$ for some $c, \beta > 0$ and data condition κ. According to the reasoning above, the algorithmic error γ should not exceed the statistical error, since otherwise we would waste parts of the available data. We thus set $\gamma = |\mathcal{D}|^{-1/2}$. In this case, the runtime becomes $\mathcal{O}(\kappa^c|\mathcal{D}|^{\beta/2}$ polylog$(|\mathcal{D}|))$, and thus, polynomial in the number of data points—an exponential speedup over classical least squares cannot be justified.

Statistical effects also affect the accuracy of the readout. A specific number of readouts is required to obtain the result of the quantum computation with a desired accuracy. Using techniques developed within the field of quantum metrology, it is often possible to achieve a precision that scales as $1/m$, where m is the number of readouts [285].

In the quantum regression setting, this implies that from state $|\hat{w}\rangle$, we can obtain $\hat{w}$ by sampling the circuit m times such that $\||\hat{w}\rangle - \hat{w}\| \leq \Omega(1/m)$. By the same reasoning as above, this means that we have to obtain $m = \sqrt{|\mathcal{D}|}$ shots for the readout in order to not discard information from our data set $\mathcal{D}$. No matter how fast our regression algorithm is, the number of readouts will be a polynomial in the number of data points.

6.2.3 Noise Robustness

NISQ devices suffer from various types of noise, e.g., measurement noise or gate errors. These uncertainties arise in addition to the general probabilistic nature of quantum computation. One might wonder why we should even care about additional stochastic effects, instead of treating them as part of the probabilistic computation itself. Correction for measurement noise exists such that there is no direct practical problem with it. However, the error induced by noisy quantum gates is notoriously harder to characterize. One reason is crosstalk in superconducting quantum processors, e.g., performing operations on two pairs of qubits simultaneously can induce an error if the qubits are physically close on the chip. While crosstalk and other sources of noise are hardware related problems, it is up to the algorithm whether slight variations during the computation will lead to a completely different result.

As an example, consider adiabatic quantum computation. AQC is a stochastic procedure—there is no guarantee that the outcome, measured after a finite amount of time, is the ground state of the target Hamiltonian. It can well be that the adiabatic evolution ends up in an excited (sub-optimal) state. The conditions under which the adiabatic process stays in the ground state until the end of the evolution are well understood [63, 286].
The key property is the Hamiltonian's spectral gap, that is, the difference between the ground state energy and the energy of the first excited state. Moreover, there is sufficient theoretical evidence that spectral gaps of random problem instances are likrly to become super-exponentially small [287, 288]. Thus, the time required for adiabatic evolution to remain in the ground state is longer than the time required for a classical brute force search through the full state space. It turns out that exponentially small gaps appear close to the end of the adiabatic evolution for large random instances of NP-complete problems. This implies that, unfortunately, adiabatic quantum optimization fails: The system gets trapped

in one of the numerous local minima. Since the adiabatic evolution of NISQ devices is subject to minor perturbations, variations can affect the spectral gap and hence alter the success probability of the computation. Thus, QUBO formulations should take this problem into account. Nevertheless, experimental results show that the QAOA algorithm seems to be more robust against exponentially small spectral gaps [289].

Another line of research addresses the formal robustness verification of quantum machine learning algorithms against unknown quantum noise. As mentioned above, noise may arise from various sources, rendering a full characterization of the noise distribution almost infeasible. In [290], the authors relate noise robustness to adversarial examples and present an analytical robustness bound to assess the robustness of quantum classification algorithms. Their results are, however, not evaluated on real quantum computing devices. At the time of writing, evaluating the noise robustness of quantum machine learning algorithms is still an open problem.

7 Conclusion and Outlook of Part I

After decades of worldwide research, artificial intelligence has finally made considerable strides with respect to capabilities and applicability. As a consequence, weakly intelligent cognitive systems are now increasingly deployed in practice and appear in more and more areas of our daily- and professional lives. An interesting general observation in this context is that accelerated progress and noteworthy recent accomplishments in this area have mainly been driven by modern large scale machine learning.

Machine learning is a branch of artificial intelligence that deals with adjusting the parameters of software agents in a training process such that they can develop cognitive capabilities and problem solving skills. Put differently, the recent performance boost in artificial intelligence is mainly due to systems which analyze large amounts of task specific training data in order to learn an intended input-output behavior. The tacit assumption behind this idea is that any real world data must have been produced by some generally unknown process or mechanism and that this process or mechanism can be modeled mathematically. This translates to the assumption that there exists a suitable parameterized function whose parameters can be automatically adjusted such that the input-output behavior of that function reflects or mimics the characteristics of the given training data.

There exist numerous possible machine learning models (mathematical formalizations of a given application scenario) as well as numerous learning algorithms (mechanisms to fit a given model to a given set of training data). Traditionally, models and algorithms for their training were chosen with respect to the task at hand, often specifically tailored towards a specific setting. However, the dominating trend over the past decade and arguably the main reason behind the success of modern machine learning has been to work with domain agnostic models and training algorithms. In particular, deep learning has proven to be tremendously successful in a wide range of computational intelligence problems.

Deep learning involves deep (and wide) artificial neural networks which are composed of millions of artificial neurons which interact over even more artificial synapses. Since such networks thus come with billions of adjustable parameters (synaptic weights and activation function bias values), they provide very flexible general models which can be trained with (variants of) the backpropagation algorithm to learn a desired cognitive skill. However, in order for this to happen reliably, deep neural networks need to be trained with vast amounts of representative training data. Due to the number of data to be processed and the number of parameters to be adjusted, the training of modern deep networks is a formidable task that requires considerable computational resources in order to happen within reasonable time. Indeed, breaking down the efforts involved in training state of the art systems such as, say, OpenAI's GPT3 for text analysis- and synthesis reveals training times of several hundred GPU years which, to be possible within a matter of days, necessitates the use of dedicated compute clusters. In other words, state of the art machine learning has reached a point where its practical feasibility and success are conditioned on access to high performance computing hardware.

Given this state of affairs regarding the computational needs of current machine learning systems, it is not surprising to find that more and more researchers are beginning to look at quantum computing as a tool to be deployed at different stages of the machine learning pipeline. The basic observation is that quantum computing, too, has made considerable strides over the past decade and is now becoming practical. While quantum computing will not eliminate the dependency on special purpose hardware, it promises significantly faster computations across a wide range of scientific or industrial applications. With respect to machine learning, the ambition is to harness potential quantum speedup especially for the training of ever more complex systems.

The advantages of quantum computing over classical digital computing are rooted in the fact that it exploits quantum mechanical phenomena for information processing.

While digital computers operate on bits which are in one and only one of two possible states, quantum computers operate on qubits. These are logical interpretations of physical two-state quantum systems which exist in a superposition of two basis states. The state space of a qubit thus consists of infinitely many states so that a qubit can carry more information than a classical bit. While the mathematics that describes the behavior of classical bits is Boolean algebra, the mathematics that describes the behavior of qubits is complex linear algebra. Qubits can be represented as two-dimensional, complex-valued unit vectors that are formed as a linear combination of two distinguished, linearly independent, orthonormal basis vectors. The squares of the norm of the coefficients of a qubit state vector are called amplitudes and have an important probabilistic interpretation. If a measurement is performed on a qubit, it will decohere, i.e. loose the property of superposition, and collapse to either one of its basis states. The probability of the measured state to be the first basis state corresponds to the squared norm of the first coefficient and the probability of measuring the qubit in the second state corresponds to the squared norm of the second coefficient. Computations involving operations on- and subsequent measurements of qubits are therefore probabilistic rather than deterministic as in the case of classical bits. However, just as classical bits can be combined into bit registers, quantum bits can also be combined into qubit registers. Adding a single qubit to a register increases the dimension of the register's state space by a factor of two and so a quantum register of n qubits exists in a superposition of 2^n basis states. Another crucial difference to classical computing is that qubits can be entangled. Whenever two or more qubits are entangled, their individual states cannot be measured separately but a measurement of an entangled qubit will also determine the (combined) state of the others.

The interplay of these phenomena gives rise to the potential supremacy of quantum computing over digital computing: on a quantum computer it is possible to work with only n qubits in order to perform computations in an 2^n dimensional space. This is of considerable interest for combinatorial optimization which deals with exponential search spaces and plays a crucial role in a branch of artificial intelligence known as problem solving, as well as in general parameter selection problems in certain machine learning techniques.

Adiabatic quantum computers such as produced by D-Wave are especially tailored towards solving an important class of combinatorial optimization problems, namely QUBOs, which occur in machine learning contexts such as data clustering, classifier boosting, or support vector machine training. The basic idea in adiabatic quantum computing is to express a given problem as an energy minimization problem and to devise an energy function which is known to attain its minima at the unknown solutions to the problem. The energy of a quantum system is characterized in terms of a Hamiltonian operator whose eigenvalue spectrum is the set of all possible outcomes when measuring the system's energy; its eigenstates reflect to which states of the system these energies correspond to. The lowest energy state of the quantum system under consideration is also called the ground state of the corresponding Hamiltonian and, if the energy function for a given problem can be translated into a Hamiltonian operator, its ground state can be found via adiabatic quantum computing. Adiabatic quantum computing exploits a phenomenon summarized in the adiabatic theorem. It states that, if a quantum system starts in the ground state of a Hamiltonian which then gradually changes towards another Hamiltonian, the system will end up in the ground state of the resulting Hamiltonian. On an adiabatic quantum computer one thus operates with two Hamiltonians, a problem independent one and the problem Hamiltonian and gradually changes the former into the latter. This general idea therefore exploits a form of quantum tunneling, i.e. a quantum mechanical principle which states that quantum systems can tunnel through energy barriers. The latter constitutes an advantage over classical energy-based optimization where optimizers may get trapped in local minima and thus may fail to find optimal solutions to a problem. Moreover, for many problems of practical importance, adiabatic quantum computers can search the solution state space polynomially faster than classically possible. While this may not sound like much, for large problems it can make the difference between tractable and intractable.

At first sight, the way of thinking required for adiabatic quantum computing may look abstract and unusual. However, problem solving by energy minimization is a well established general paradigm in classical machine learning. In fact, the well understood Hopfield networks, a venerable type of neural networks, can be seen as a classical or digital analogue to adiabatic quantum computing. This is to say that anything that can be accomplished by running a classical Hopfield network on a digital computer can, in principle, also be accomplished on an adiabatic quantum computer. In this sense, the conceptual gap between the classical paradigm of Hopfield neural networks and adiabatic quantum computing is therefore rather narrow.

Quantum gate computing manipulates qubits in a manner that more closely resembles classical digital computing. On the hardware level, digital computers process information by manipulating sets of bits via atomic logical operations which are implemented in terms of so-called gates. In quantum gate computing, corresponding operations on qubits are realized via the application of quantum mechanical operators. Mathematically, these are unitary transformations so that any operation on qubits must be unitary too. Such operators typically act on one or two qubits at a time but can be sequenced or executed in parallel to form more complicated operations on sets of qubits. In analogy to classical computing, complex computational units that are composed of individual quantum gates are called quantum circuits. From a logical point of view, the fundamental problem in quantum gate computing is therefore to devise quantum circuits that show an intended input/output behavior. This is generally a non-trivial problem and it is interesting to note that classical machine learning is increasingly seen as tool for quantum circuit design.

What makes quantum gate computing attractive from the point of view of machine learning is that, mathematically, quantum gate computing is nothing but applied complex linear algebra. Due to mechanisms such as superposition or entanglement, however, the linear algebraic operations that occur in quantum gate computing implicitly act on state spaces that are exponentially larger than those considered in classical computing. Since many common machine learning tasks involve linear algebraic operations on large amounts of high dimensional data vectors, it seems auspicious to try to encode such data in qubit state spaces and to try to leverage quantum advantages when computing with such states. Here, the general expectation is that quantum speedup will considerably accelerate corresponding learning processes or even allow to tackle hitherto intractable problems.

Indeed, in the words of Aaronson, there has recently been a “quantum machine learning mini-revolution” and the number of scientific reports on quantum circuits for certain general problems in machine learning has grown considerably over the past decades. Problems which have been considered in this context include basic linear algebra routines, regression, or classification. With respect to the latter, there exist different paradigms so that it is no surprise to find numerous proposals of quantum circuits for nearest neighbor classifiers, basic linear classifiers (perceptrons), ensemble classifiers, support vector machines, and neural networks.

Present day ideas for how to realize quantum machine learning routines on quantum gate computers commonly involve variational quantum computing algorithms or hybrid quantum-classical methods. These consider parameterized quantum circuits which may be designed manually or automatically but in either case consist of tunable quantum gates. The problem solved by hybrid quantum-classical methods therefore is the problem of adjusting the parameters of individual gates such that the circuit as a whole performs the desired computation with high probability. The basic structure of variational quantum algorithms consist of an outer loop run on a digital computer which manages the current estimates of the sought after parameters. In each iteration, these parameters are used to setup computations on a quantum computer whose outcomes are then measured. Given these measurement results, classical optimization techniques are then used to estimate improved parameters and the whole processes is iterated until the quantum circuit shows the desired input/output behavior within a given tolerance.

Given the technical capabilities of existing, present day quantum computers, variational- or hybrid quantum-classical algorithms are appealing because they have been found to considerably reduce the quantum computing resources (circuit depth, coherence time, qubit counts) that are required to successfully run a quantum machine learning method. Moreover, researchers interested in developing quantum neural networks often even consider parameterized quantum circuits as a quantum computing analogue of classical deep neural networks. This is likely because deep neural networks as well as parameterized quantum circuits consist of basic computational units arranged in layered structures and because both involve classical optimization algorithms for parameter adjustment. However, one has to be careful with such analogies since there also are crucial differences. First of all, quantum gates in a quantum circuit realize unitary operators rather than non-linear functions as in the case of neural networks. Non-linear computations are vital for the general problem solving capabilities of neural networks but they can not be realized through unitary operators alone. Non-linear quantum computations either involve the generous use of ancilla qubits or state measurements. Second of all, internal states of a quantum circuit cannot be read out without destroying their quantum coherence, i.e. without loosing quantum aspects such as superposition. Classical neural network training based on error backpropagation does therefore not apply to parameterized quantum circuits and the use of variational- or hybrid quantum-classical algorithms for circuit optimization currently constitutes the only viable approach to this problem.

The ever faster growing literature on quantum machine learning and the many success stories reported therein seem to suggest that the application of quantum computing algorithms to computational intelligence problems could soon break into the mainstream. However, in the present era of noisy intermediate-scale quantum (NISQ) computing, overly enthusiastic expectation still need to be reigned in. This has mainly to do with the technical capabilities of present day quantum computers which often differ from the assumptions made by quantum algorithm developers.

First of all, quantum algorithm design considers properties of logical qubits rather than those of physical qubits. The former are the basic building blocks on which quantum algorithms operate, the latter are physical devices inside of a quantum computer which behave like two-state quantum systems. From the point of view of algorithm design, the focus on logical qubits is reasonable and mimics levels of abstraction in modern software development where most programmers need not worry about hardware details. Current quantum computers, however, have not yet reached the same level of maturity as digital computers but still exhibit certain limitations. As of now, pure logical qubits are therefore an idealization since they abstract away shortcomings of physical qubits in present day NISQ devices. These typically contain less than a hundred qubits, often suffer from limited coherence times, and are susceptible to low fault-tolerance due to internal fluctuations or measurement noise. Indeed, it still is challenging to technically create and manipulate quantum states and to maintain their quantum mechanical properties over longer periods of time. It is rather marvelous that this has become possible at all and experts expect that continuing technological progress will lead to better and more powerful devices. Lack of fault tolerance is a critical issue and a technological milestone will be the implementation of quantum error correction mechanisms similar to those used in digital computing. Here, it is interesting to note that errors often result from the application of less robust quantum gates. As of this writing, only rather small quantum circuits work really stable. Since adiabatic quantum computers do not involve quantum gate computations, they can more reliably manipulate larger (physical) qubit systems. At the same time, their use is currently restricted to rather specific energy minimization problems so that present day devices are not as universal as quantum gate computers. Even though both paradigms are theoretically equivalent, the emulation of quantum circuits on an adiabatic quantum computer would require qubit connectivity structures that have not yet been realized.

While presently realizable qubit counts, qubit connectivity, dimensionality of quantum gates, and quantum circuit depths impose practical restrictions on the kind of quantum algorithms that can run successfully on current NISQ devices, there are further limitations regarding the feasibility of certain logical quantum computing concepts. We therefore emphasize the following additional points.

Second of all, it is often not immediately clear how to encode classical data such that it can be processed on quantum computers; neither may it be obvious how to decode measured qubit states into classical representations that would allow for meaningful downstream processing. This input-output problem is often abstracted away by quantum algorithm designers which, in turn, may have dire consequences with respect to claims about quantum speedup. For instance, a quantum algorithm operating on a quantum encoding of classical data might be exponentially faster than its classical counterpart operating on the classical data. However, if the effort for preparing quantum states that encode the classical data is itself exponential, then the apparent quantum advantage vanishes. Here, it is noticeable that designers of quantum algorithms often simply hypothesize universal quantum computers. Among others, these machines are supposed to come with quantum random access memories (QRAMs) which, similar to their digital counterparts, can hold data for processing. However, given present day technologies, such QRAMs are not yet possible and it is even questionable if they will ever exist in their truest sense. This is due to another quantum mechanical principle, the no-cloning theorem, which states that it is impossible to create independent identical copies of arbitrary quantum states. To the best of our current knowledge, and unless reliable quantum error correction becomes available, a QRAM would thus only allow for approximate repeated access to quantum states for processing. Quantum state decoding, too, can diminish quantum advantages. Even if a quantum algorithm can produce a quantum state representation for a sought after solution much faster than classically possible, the effort for measuring and reading the resulting state into classical memory could still be so substantial that any advantage disappears. Before claims as to the superiority of quantum algorithms can be made, it is therefore pivotal to factor in efforts for state preparation or measurements and to match these against pre- and post-processing efforts of efficient classical algorithms.

Third of all, as of now quantum computing is essentially bit level computing. That is, present day quantum algorithm design deals with the design of problem specific quantum circuits or problem specific energy functions. Contrary to classical computing, there are no abstract data structures, such as linked lists, binary trees, or heaps. Neither are there or control structures such as `if-then-else`-statements or `for`- or `while`-loops known from higher level programming languages and common classical programming patterns such as the use of variables, too, are not immediately possible on present day quantum computers. This is challenging in so far as it suggests that certain existing machine learning algorithms build around such constructs may, for the foreseeable future, not be realized on quantum computers. While there are increased efforts towards quantum compilers which automatically translate higher level programs into quantum circuits, these are still in their infancy. Also, while high level application programmings interfaces (APIs) for quantum computing are increasingly available (e.g. Qiskit, Cirq, Forest, PennyLane), it is important to note that these are mainly tools for setting up quantum computing processes. That is, users of these tools still have to think on the linear algebraic or “state and operator” level of quantum computing and use the API only to implement vectors, matrices and their interplay. Another caveat with respect to these APIs is that they often allow for efficient digital simulations of quantum information processing. While this is undoubtedly helpful in the design stage of algorithm design, it can also be misleading as the simulated quantum processors typically are universal quantum processors. This, in turn, means that a quantum machine learning algorithm that works on a simulated quantum computer may not work on a currently existing physical quantum computer. A final caveat with regard to quantum algorithms for machine learning is that quantum computations which involve measurement steps are inherently probabilistic. This is of course well known to quantum computing practitioners but may be overlooked by novices. Any quantum computation must therefore be repeated several times and any result obtained this way has to be interpreted in terms of expectations rather than in terms of deterministic outcomes.

Fourth of all, while machine learning is a comparatively new scientific discipline, quantum machine learning is even newer. This has consequences for the best practices and standards in the field. To clarify what this may mean, we note that, in its first couple of decades, classical machine learning has gone through a verifiability and reproducibility crisis. Practical results tended to be reported, without disclosing implementation details, data collections or processing protocols, or experimental procedures in detail. Regarding the validity of claimed capabilities of a reported method, such omissions can make a considerable difference. It is, for instance, pivotal that training and testing of a machine learning system happen on independent data sets, because a low error on the training data does not imply that the trained system can generalize well. On the contrary, an exceedingly good performance on the training data is often a symptom of overfitting. Over time, these issues have been recognized by the machine learning community and, as a consequence, present practice for scientific publication is to require authors to provide their code, data, and experimental protocols. In the new field of quantum machine learning, this is not yet the case. Indeed, reading the corresponding scientific literature, it is noticeable that crucial details as to how a practical result has been obtained are often missing. In many reports on practical quantum machine learning, it often not even recognizable if, say, rigorous evaluation practices have been followed so that one may argue that present day quantum machine learning is experiencing a reproducibility crisis period. For now, this may be acceptable as the field is in its nascent phase; it is, however, important to keep in mind that the good performance of some of the currently reported methods may not scale or generalize to large or different application settings.

While all these caveats may dampen enthusiasm for the prospect of quantum machine learning, recent technological progress has been substantial enough to merit serious engagement with the topic. In other words, although quantum computers and quantum machine learning algorithms are not yet mature enough to impact the way machine learning happens in practice, it currently seems reasonable to expect that—due to considerable investments by institutional and industrial stakeholders—the underlying technology will continue to develop and improve ever more quickly. This, in turn, may soon lead to practically viable solutions and cause hitherto unexpected developments and disruptions. A foresight process which fathoms potential benefits and risks of quantum machine learning therefore seems appropriate.

In particular, potential risks related to the use of quantum machine learning have not yet received the same amount of attention as its benefits and corresponding reports are scarce. In other words, while ethics, reliability, trustworthiness, and safety of classical machine learning have by now been recognized as important topics, quantum machine learning has not yet been scrutinized in these regards. However, given the expected impact of quantum machine learning on capabilities and utilizability of artificial cognitive systems, it seems appropriate to assess potential security issues related to quantum machine learning. In upcoming reports, we will therefore investigate questions pertaining to the reliability or vulnerability of quantum machine learning systems. Further considerations include whether or not quantum machine learning allows for new kinds of attacks on- or new defense mechanisms for critical digital infrastructures. The general goal will be to assess quantum machine learning from the point of view of cybersecurity and to determine what kind of measures, if any, will be required in this regard.

Glossary for Part I

Adiabatic quantum computing (AQC) A quantum computing paradigm that is based on the adiabatic theorem and particularly tailored towards solving QUBOs; in adiabatic quantum computing, a qubit system gradually evolves from the ground state of a problem independent beginning Hamiltonian to the ground state of a Hamiltonian which models a given problem; the method thus uses quantum annealing or quantum tunneling for problem solving.

Adiabatic quantum optimization (AQO) A synonym for adiabatic quantum computing.

Adiabatic theorem A mathematical formulation of the quantum mechanical principle that, if a quantum system starts in the ground state of a Hamiltonian operator which then gradually changes for a period of time, the system will end up in the ground state of the resulting Hamiltonian.

Application phase (of an ML system) The final stage in the practical development of a machine learning system; once the system has been trained and tested and was found to perform robustly and reliably, it can be deployed in practical applications.

Artificial Intelligence (AI) A branch of computer science concerned with the design, development and deployment of technical systems or software agents that have cognitive capabilities such as text- or image understanding, planning, and decision making.

Backpropagation A very widely used algorithm for the training of (feed-forward) neural networks; given an appropriate loss function, the algorithm essentially applies the chain rule of differentiation to adjust the parameters of all neurons in all layers of a neural net; put differently, the method performs gradient descend in a highly non-linear error landscape; there exist numerous variants of the original version of the algorithm, for instance, to automatically determine optimal step sizes for the descent procedure.

Basis states States of quantum mechanical systems are described as linear combinations over basis states; for instance, the state of a qubit is represented as a two-dimensional, complex-valued unit vector that is a linear combination of two distinguished, linearly independent, orthonormal basis states.

Beginning Hamiltonian A problem independent Hamiltonian operator used in adiabatic quantum computing; the beginning Hamiltonian is usually chosen such that its ground state can be easily prepared and used as the initial state for the adiabatic problem solving process.

Computational basis A basis system used to express the state of a qubit system; the state of a system of qubits corresponds to a vector in a complex Hilbert space and the basis used to (numerically) express the entries of this vector can be chosen arbitrarily; the computational basis is often the most "natural" choice, i.e. a basis that suits the problem at hand or in which state vectors have a simple form.

Decoding problem The problem of measuring or reading a quantum state which results from a quantum computation; quantum computing results usually need to be transferred to digital computers in order to enable further processing or analysis; if this transfer cannot be done efficiently, advantages due to quantum speedup can be lost.

Decoherence A term referring to the loss of quantum coherence; if a quantum systems decoheres, it loses its quantum properties; decoherence of qubit happens through measurement or (unwanted) interactions with the outside world; qubit systems have thus to be kept as isolated as possible when performing quantum computations; if a qubit decoheres, it collapses to one of its basis states and henceforth behaves like a classical bit.

Digital annealer A specialized digital computing device tailored towards solving QUBOs; digital annealers classically emulate the workings of adiabatic quantum computers; this requires efficient (hardware) implementations of classical optimization algorithms; compared to present day adiabatic quantum computers, digital annealers are much cheaper and much more energy efficient.

Eager learner A machine learning system that trains a model in an offline training phase and considers a complete existing training data set for re-training and rather than only newly added data.

Encoding problem The problem of preparing quantum states that represent classical data; if classical data cannot efficiently be encoded in terms of quantum states, computational advantages due to quantum speedup can be lost.

Ensemble learning Ensemble Learning methods combine different, typically simple (machine learning) models into a single stronger model that achieves better results than the individual member of the ensemble. Ensemble Learning allows, for example, for an average value from the results of these different models.

Entangled state In quantum computing, this term refers to a state in which a system of qubits can exists such that their individual states cannot be measured separately; rather, a measurement of an entangled qubit will also determine the (combined) state of the others; entanglement is a quantum mechanical phenomenon for which there is no classical analogue and which is essential for the inner workings of quantum computers.

Federated learning In federated learning, each participant retains their own data and contributes to a common learning process in which the system as a whole benefits from the capabilities of each contributor.

Generalization The capability of a trained machine learning system to generalize the predictive performance it acquired from analyzing training data to previously unseen test data.

Ground state A term used to refer to the lowest energy state a quantum mechanical system can be in; mathematically the lowest energy of a quantum system corresponds to the bottom eigenvalue of its Hamiltonian operator and the system's ground state is given by the corresponding eigenvector.

Hamiltonian operator A quantum mechanical operator acting on the state vectors of quantum mechanical systems; Hamiltonians feature prominently in Schrödinger equations; their eigenvalues represent possible energy levels the corresponding quantum system can be measured in.

Kernel machine A machine learning term used to refer to a wide class of machine learning models; kernel machines are machine learning models that make use of the kernel trick when processing data.

Kernel trick A computational method used in kernel machines which allows for applying inherently linear methods to non-linear machine learning problems. Invoking the kernel trick involves two steps: 1) to rewrite a given machine learning method such that any occurrence of data vectors is in form of inner products and 2) to replace any computation of an inner product by the computation of an appropriate kernel function.

Lazy learner A machine learning system that simply stores data and delays modeling until asked to make predictions; a simple example for this paradigm is a nearest neighbor classifier.

Logical qubit A mathematical idealization of a physical two-state quantum system; logical qubits exist in a superposition of two basis states and can therefore represent more information than classical bits; they form the basic units in quantum computing.

Loss function A criterion to measure how well a machine learning model and its current choice of parameters represent a given set of data; quality measurement is done by assigning to each prediction of the model the loss that occurs when a prediction deviates from the correct prediction; the more predictions deviate from the ground truth training data, the higher the output of the loss function.

Machine Learning (ML) A branch of artificial intelligence that deals with adjusting the parameters of technical systems or software agents in a training process such that they can develop cognitive capabilities and problem solving skills.

Model class A machine learning term used to refer to a parameterized family of functions; once a model class has been chosen for a machine learning problem, the task is to determine that model within the class that provides the best fit to a given set of training data; adjusting the model to the data at hand happens through machine learning algorithms which tune its parameters such that the model is in good agreement with the data; often, this happens by means of minimizing a loss function.

No-cloning theorem A quantum mechanical principle that states that it is impossible to create independent identical copies of arbitrary quantum states.

Noisy intermediate-scale quantum device (NISQ) A term used to describe existing, present day quantum computers which contain less than a hundred physical qubits and still suffer from limited coherence times and low fault-tolerance.

Oracle A kind of function posited or required in many quantum computing algorithms; oracle functions can quickly verify if a supposed solution to a problem really is a solution; for instance, while the problem of finding the prime factors of 42 is somewhat difficult, it is easy to verify that 2, 3, and 7 are the solution; by the same token, it is also easy to verify that, say, 5 and 13 are not.

Overfitting A type of machine learning modelling error that occurs when a model corresponds too closely to a given data set and is therefore likely to produce very different results for (slightly) different data samples.

Parameterized quantum gate A kind of quantum gate that requires numeric parameters to define its actual function; parameterized quantum gates are of pivotal importance for variational quantum algorithms and for quantum machine learning; they allow for representing machine learning models via a single quantum circuit and for the use of optimization techniques to adjust the input/output behavior of the circuit such is agrees with given task specific data.

Phase gate A single qubit quantum gate that modifies or shifts the phase of the quantum state of a qubit.

Physical qubit A physical realization of a logical qubit, namely a device or apparatus that behaves like a two-state quantum system and forms a component of the hardware of a quantum computing system.

Problem Hamiltonian A Hamiltonian operator used in adiabatic quantum computing; the problem Hamiltonian is designed in such a manner that its ground states represent the sought after solution to a given (combinatorial optimization) problem; in adiabatic quantum computing, a qubits system is prepared in the ground state of a problem independent beginning Hamiltonian which then gradually changes towards the problem Hamiltonian; this will cause the qubit system to end up in a ground state of the latter and thus solve the problem at hand. The design of problem Hamiltonians requires experience with (re)formulating QUBOs.

Quadratic unconstrained binary optimization (QUBO) A kind of combinatorial optimization problem where the decision variables can only assume two values (typically either 0 and 1 or -1 and $+1$); many practically important subset selection or set bi-partition problem can be written as QUBOs but their solution is classically difficult as they are generally NP-hard.

Quantum approximate optimization algorithm (QAOA) A variational quantum algorithm in which parameterized quantum gates are optimized to approximate adiabatic quantum computations.

Quantum bit (qubit) A two-state quantum system; a qubit exists in a superposition of two basis states and can therefore represent more information than a classical bit; qubits are the basic unit of information in quantum computing or quantum information processing and are modeled in terms of vectors in a complex Hilbert space; one often distinguishes between logical qubits (which represent the mathematical essence of two-state quantum systems) and physical qubits (which are physical instances or realizations of two-state quantum systems); in other words, while there exists different physical manifestations of qubits, their essential (mathematical) properties are the same.

Quantum computing A computational paradigm that harnesses the principles of quantum mechanics for information processing; exploiting quantum mechanical phenomena such as superposition or entanglement offers great computational power but also requires a different kind of algorithmic thinking than in classical digital computing.

Quantum gate computing A quantum computing paradigm sometimes also referred to as the quantum circuit model of computation; in quantum gate computing, operations on qubits are realized via the application of quantum mechanical operators which are physically implemented as quantum gates; in analogy to classical computing, complex computational units which are composed of individual quantum gates are called quantum circuits.

Quantum gate A basic computational unit within a quantum circuit; quantum gates are used to perform quantum mechanical operations on individual qubits or on systems of qubits.

Quantum information processing A scientific discipline that deals with theory and practice of systems which process quantum information; an important aspect of quantum information processing is quantum computing; however, quantum information processing is sometimes seen to be more comprehensive than quantum computing because it also involves topics such as quantum communication or quantum sensing.

Quantum inspired computing A term used mainly to describe classical algorithms whose design or operating principles are inspired by- or try to emulate quantum mechanical phenomena; for instance, a digital annealer digitally emulates the inner workings of an adiabatic quantum computer.

Quantum machine learning A term mainly used to refer to the idea of incorporating quantum computing routines at different stages of the machine learning pipeline; in a much broader sense, the term is also used to refer to quantum inspired classical algorithms for classical data analysis, genuine quantum algorithms for classical data analysis, classical algorithms for quantum data analysis, and quantum algorithms for quantum data analysis.

Quantum supremacy A term used to describe the fact that there exist problems which a quantum computer can solve but a classical computer can not in feasible time; in order to exhibit quantum supremacy with respect to a given problem, a quantum algorithm has to provide super-polynomial speedup over the best possible classical algorithm.

Quantum tunneling A quantum mechanical principle that describes the fact that quantum systems can tunnel through energy barriers; quantum annealing or adiabatic quantum computing exploit this phenomenon.

Regularization A mathematical technique or concept designed to prevent machine learning algorithms from overfitting by imposing constraints on model parameters that reduce the variance of a model.

Reinforcement learning A machine learning paradigm tailored to the problem of learning "what to do when" in order to achieve a long term goal; reinforcement learning typically considers states an agent can be in, actions an agent can perform to transit to another state, and delayed rewards an agent has to maximize.

Superposition A quantum mechanical principle which states that the sum (superposition) of two or more quantum states is another valid quantum state; superposition is often interpreted in the sense that a quantum mechanical system can be in many different states simultaneously. Upon measurement, the system will collapse to one of the superposed states.

Supervised learning A machine learning paradigm where machine learning models are trained on annotated training data; the goal is to adjust the model parameters such that the model is able to map the given input data to the given output data with high fidelity; importantly, models trained in a supervised manner have to be evaluated on independent test data in order to verify that they can generalize to situations not contained in the training data; a common example of a supervised learning problem is classifier training.

Test phase (of an ML system) A stage in the practical development of a machine learning system; given problem specific test data and a trained model, the model is evaluated on the test data using an appropriate performance measure; it is pivotal that the test data are independent of the data used to train the model, otherwise issues such as overfitting cannot be detected.

Topological quantum computing (TQC) A still mainly theoretical concept for how to realize physical qubits based on the idea of working with quasi-particles called anyons.

Training phase (of an ML system) A stage in the practical development of a machine learning system; given problem specific training data and a parameterized mathematical model, the model parameters are adjusted automatically such that the model matches the training data to the best extend possible; often, optimal model parameters are estimated by means of minimizing a loss function.

Two-state quantum system A quantum mechanical system whose state space consists of infinitely many states which can be represented as two-dimensional, complex-valued unit vectors that are linear combinations of two distinguished, linearly independent, orthonormal basis states. Well known real world examples of two-state quantum systems include the polarization of a photon (with basis states vertical or horizontal) or the spin of an electron (with basis states up or down).

Unsupervised learning A machine learning paradigm that aims to detect and uncover latent or inherent structures within a given data set of unlabeled data; a common example of an unsupervised learning problem is data clustering.

Variational quantum algorithm An algorithm that combines quantum computations with classical computations in an iterative feedback loop; variational quantum algorithms or hybrid quantum-classical algorithms are used to adjust the parameters of a parameterized quantum circuit such that it shows an intended input/output behavior within a given tolerance; a variational quantum algorithm involves an outer loop run on a digital computer which manages the current parameter estimates, in each iteration, these parameters are used to setup computations on a quantum computer whose outcomes are measured, measurements are then used in classical optimization techniques to estimate improved parameters and the whole processes iterates until convergence.

Variational quantum eigensolver (VQE) A variational quantum algorithm for estimating the bottom eigenvalues of Hamiltonian operators.

Part II

Security within Quantum Machine Learning

8 Security in the Application of Quantum Machine Learning

This part takes a closer look at current quantum machine learning algorithms, systems, and technologies from the point of view of cyber security. This includes potential attacks, as well as defense mechanism.

To begin with, we therefore clarify the sense in which the term quantum machine learning will be understood and which peculiar characteristics quantum machine learning has. That is, we briefly recall the following:

Machine learning (ML) is a branch of artificial intelligence (AI) that deals with adjusting the parameters of software systems such that they develop cognitive skills. This requires appropriate mathematical models for the skill to be acquired and involves a training phase in which intended input-output behaviors are learned from examples. While there exist numerous ML models (mathematical representations of learning systems) as well as numerous ML algorithms (mechanisms to adjust model parameters), the currently dominating trend is to work with domain agnostic methods such as deep neural networks. These have proven remarkably successful in many applications, however, as they might involve billions of adjustable parameters, training involves extensive computations and thus training times are significant. More and more researchers and practitioners are therefore beginning to consider quantum computing as a tool to accelerate machine learning.

Quantum computing exploits quantum mechanical phenomena such as superposition and entanglement for information processing. Working with quantum bits (qubits) offers great computational power but differs considerably from conventional computing since it involves complex linear algebra and properties of reversible unitary operators. Systems of n logical qubits can be modeled as complex-valued vectors which represent superpositions of 2^n basis states. Measuring such a system will cause it to decohere, that is to lose its quantum properties, and to probabilistically collapse to one of its basis states. Another important difference to classical computing is that qubits can be entangled such that their individual states cannot be measured separately. That is, whenever two or more qubits are entangled, a measurement of one of them also decides the (combined) state of the others. The interplay of all of these phenomena gives rise to the potential supremacy of quantum computing.

Seen from the perspective of classical ML, *quantum machine learning* (QML) refers to the idea of using quantum computing algorithms for computationally demanding learning tasks. Put differently, QML is the idea of processing classical data (or, in potential future applications, even quantum data) on quantum devices to train intelligent systems. The expectation is that quantum speedup will accelerate learning, or even allow for tackling problems which are beyond reach on (state of the art) classical computing devices. Indeed, the quickly growing literature on QML and the success stories it reports seem to suggest that quantum computing may soon disrupt AI technologies. Yet, it is important to note that current quantum computers are noisy intermediate-scale quantum (NISQ) devices. Any near term application of QML will therefore have to pay attention to their technical limitations.

First of all, quantum algorithms consider logical qubits rather than physical qubits. The former are the fundamental building blocks of quantum computing, the latter are physical devices inside a quantum computer. As of now, logical qubits are still an idealization which abstracts away shortcomings of NISQ devices. Indeed, current NISQ devices only realize about a hundred logical qubits and suffer from limited coherence times and low fault-tolerance due to internal fluctuations or measurement noise.

Second of all, it is often not yet clear how to practically encode classical data into quantum state representations; neither may it be clear how to decode quantum measurements into classical representations for meaningful downstream processing. This input-output problem may cause practically achievable quantum speedup to come to naught. It is indeed noticeable that the developers of quantum algorithm often hypothesize universal quantum computers which, given present day technologies, are not yet possible.

Third of all, present day quantum computing is essentially bit level computing and mainly deals with the design of problem-specific quantum circuits or energy functions. Contrary to classical computing, high level abstract quantum data structures and high level quantum control structures are still missing.

Fourth of all, QML is an emerging discipline which still needs to develop best practices and standards. Indeed, current scientific reports on QML often seem to omit details as to how practical results were obtained and how performance evaluations were carried out so that it is not always clear if rigorous (evaluation) protocols have been applied. This puts into question some reported good QML performances, especially as it may cause exaggerated expectations or confidence in the capabilities of present day QML.

Individually and as a whole, these issues could quickly develop into potential vulnerabilities of QML and raise questions as to the security of QML in real world applications. Next, we therefore fathom the possibility of malicious attacks on QML systems as well as potential defense mechanisms or countermeasures.

Given current technical quantum computing capabilities, this analysis will necessarily involve a foresight perspective. Crucial questions are whether known vulnerabilities of classical ML systems systematically transfer to QML systems, or whether one can conceive of novel points of attack that are specific to QML. As it is common in security contexts, we will frequently specifically consider the point of view of end-users who apply machine learning on quantum computers and ask which well known or novel aspects they would need to take into consideration. That is, we will investigate how security concepts or strategies may need to be adapted when transiting from classical ML solutions to potential QML solutions.

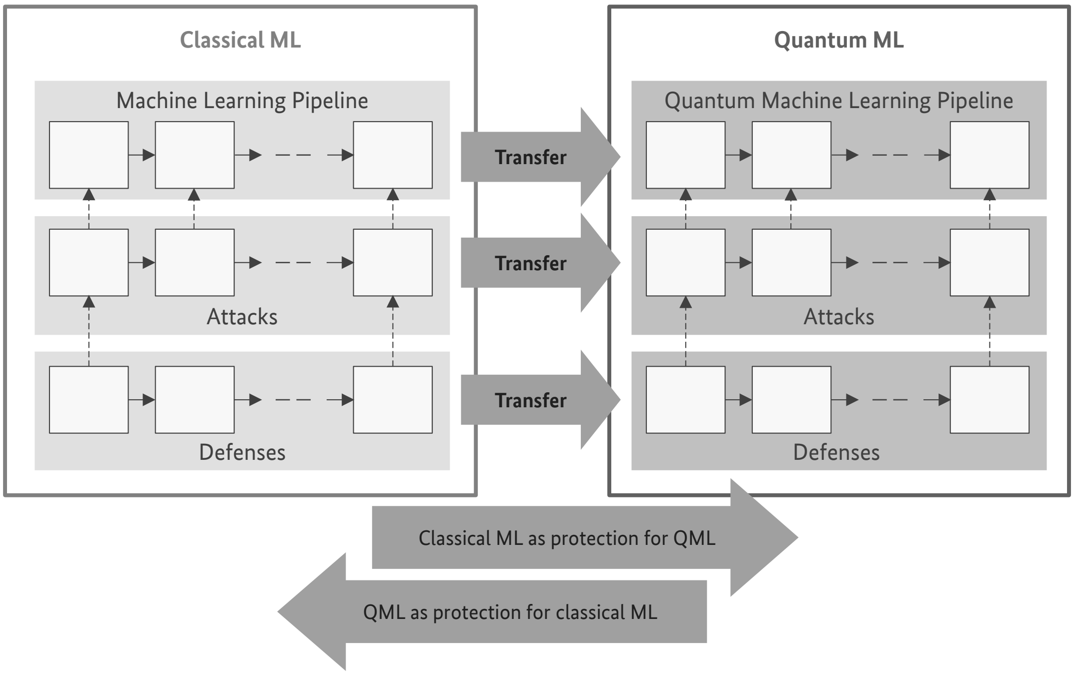

Classical ML
Machine Learning Pipeline
Attacks
Defenses
Transfer
Transfer
Transfer
Quantum ML
Quantum Machine Learning Pipeline
Attacks
Defenses
Classical ML as protection for QML
QML as protection for classical ML

Figure 5: Transfer from classical ML to QML

Section 9 will particularly address the questions of *"What could make QML especially vulnerable?"* and *"How does the QML lifecycle differ from the classical ML lifecycle?"*. In line with current literature, the central element of analysis is a transfer of ML properties and vulnerabilities into the world of QML, as shown in figure 5. The following sections 10 and 11 then additionally delineate dimensions and types of attacks and discuss *"What new attack points and scenarios result for QML?"* Especially the latter gives rise to several more specific questions on which we elaborate by surveying the current literature.

In particular, we investigate what kind of manipulations could perturb the performance of a quantum classifier and ask if changes in the training data make these systems more or less susceptible to mis-classifications than their classical counterparts. Put differently, we ask about the feasibility of *data poisoning attacks* against QML systems and whether these may require smaller or larger poisoning budgets.

Other kinds of attacks considered include *privacy attacks* and *model stealing attacks*. Here, we ask whether information about training data is easier or more difficult to extract from QML models than from traditional ML models and whether the functionality of a QML model is easier or more difficult to extract than that of conventional ML models. Closely related questions consider whether attackers can build substitute- or surrogate-models of a system. Furthermore, we consider *adversarial attacks* and ask whether adversarial examples are easier or more difficult to transfer between QML models than between conventional ML models.

Considering (federated) learning scenarios where QML takes place at a remote location, we investigate the possibility of attacks in which external third parties can participate in an unauthorized manner or even disrupt ongoing learning processes. Here, we ask, for example, whether there are protocols for secure QML taking place remotely or distributed. Last but not least, we look at the possibility of *side-channel attacks* on QML systems.

Generally, all these questions pertain to the aspect of robustness of quantum classification or regression and the conditions under which different QML models are robust enough for practical use in real-world applications. Put differently, we ask whether different QML model classes could systematically be hardened against different types of attacks and—if so, then in what way. This then raises questions as to the costs of defensive measures and, crucially, whether additional resources potentially required for defensive measures may annihilate the expected advantages of quantum computing for machine learning.

Finally, in Section 12, we look at further defense strategies against attacks on QML systems. In preparation of future work, we discuss whether, in the context of pure application security, QML methods can possibly serve as a protection means against attacks on QML systems.

9 The Quantum Machine Learning Pipeline

9.1 The Software Engineering Lifecycle in Quantum Computing

A large portion of coding for quantum computers currently still happens on a level very close to the hardware—comparable to machine code and assembly language in the classical world. Further abstraction layers, while reducing complexity for the coder, add computational overhead. In the case of quantum computing, this would be reflected in the use of additional gates as well as increased circuit depth. That means that the more limited the hardware is, the higher the benefits of coding with less abstraction levels become. The current state of quantum coding mirrors the early stages of coding on classical hardware, which has seen multiple dramatic improvement cycles in abstraction, and thus coding ease. Clearly, a similar development will sooner or later happen in quantum computing.

However, that means that the current (classical) software engineering lifecycle is not fully suited for development of quantum code. Weder et al. [291] present a lifecycle of quantum software development consisting of ten phases, which serves to better understand the development process of quantum applications. It also delivers insights into the differences between classical and quantum software engineering. The authors take into account the fact that current quantum applications are hybrid solutions composed of classical and quantum components and consider these in their model. This distinguishes their lifecycle from previous ones, which always focus on only one aspect, quantum or classical. A simplified visualization of the lifecycle is presented in figure 6:

1. Quantum-Classical Splitting: separation of parts that use quantum computation
2. Hardware Independent Implementation: a first implementation
3. Quantum Circuit Enrichment: initialization and other details are added
4. Hardware-Independent Optimization: a first optimization
5. Quantum Hardware Selection: selection of suitable quantum systems
6. Readout-Error Mitigation Preparation: reduction of the effects of errors
7. Compilation & Hardware-Dependent Optimization: further optimization is now possible
8. Integration: deployment to the quantum (and classical) systems
9. Execution: the computation is performed
10. Result Analysis: before the next iteration, the output is evaluated

Arais et al. [292] also point out that advances in quantum computing and thus the increasingly relevant fields of quantum software engineering not only open up new possibilities, but also new attack areas that need to be taken into account at the very beginning of the design process. In particular, quantum software engineering requires new so-called quantum programming languages (QPL)—the security aspects of which have yet to be in-

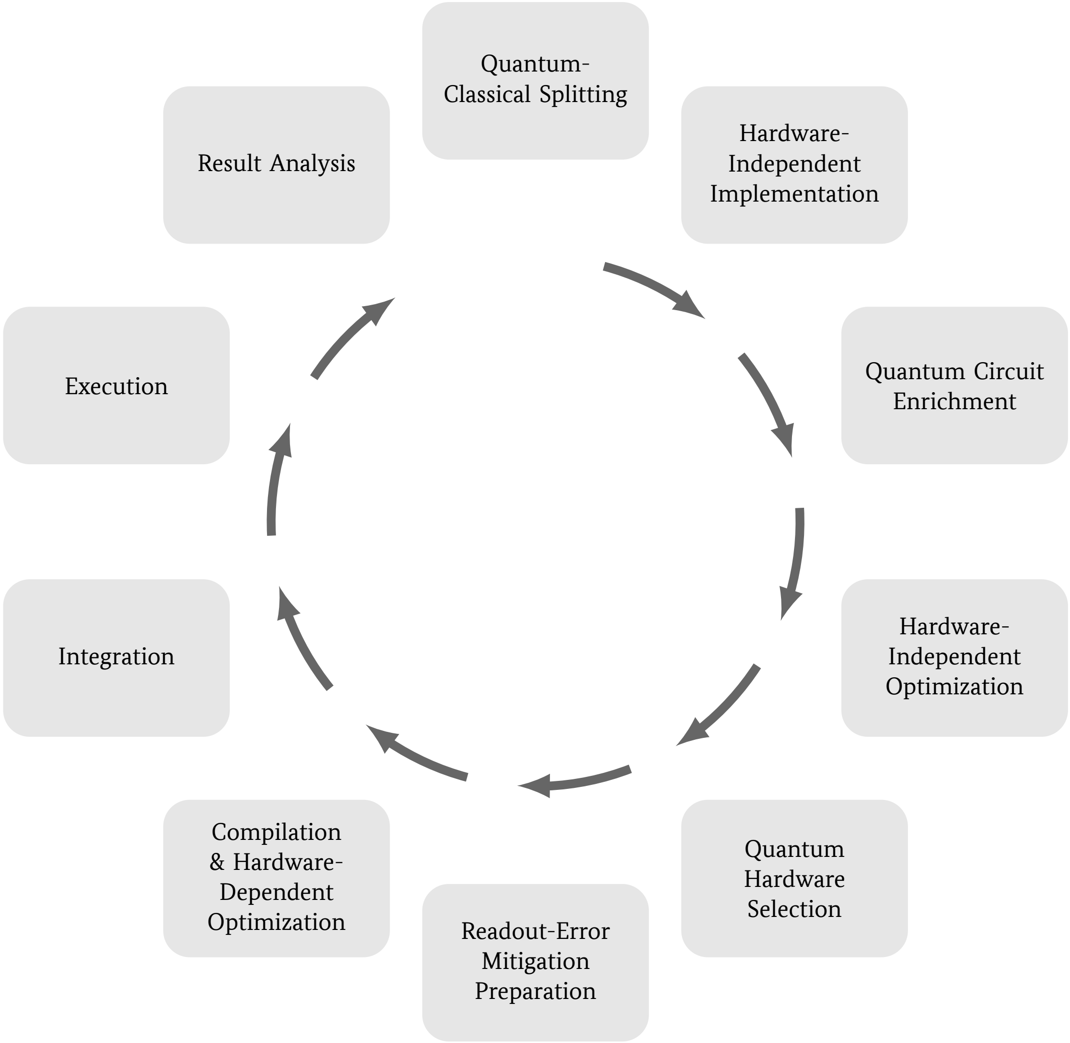

Quantum-Classical Splitting
Hardware-Independent Implementation
Quantum Circuit Enrichment
Hardware-Independent Optimization
Quantum Hardware Selection
Readout-Error Mitigation Preparation
Compilation & Hardware-Dependent Optimization
Integration
Execution
Result Analysis

Figure 6: The Quantum Software Lifecycle presented in [291]

vestigated. For example, the problem arises that classical methods for code analysis cannot be simply applied to QPL and that current QPL require programmers to have good knowledge of quantum computing.

An overview prepared by the Federal Office for Information Security (BSI) examines how ML technologies could be used in a safer, more robust and more traceable way [293]. The authors list current problems within ML, improvement measures and present the need for action. In particular, they highlight fields with potentially critical effects, such as autonomous driving, facial recognition, or the analysis of medical data. There is an emphasis on the need to determine and evolve standards, testing methods and testing criteria, as well as on the exploration of effective countermeasures against AI-specific attacks. In addition, the authors point out the relevance of exploring methods of transparency and explainability.

The researchers from BSI additionally establish a first collection of test areas and specific test criteria for an evaluation of the security of an ML application throughout its lifecycle in their “AI Cloud Service Compliance Criteria Catalogue (AIC4)” [294].

9.2 Attack Surfaces in Quantum Computing

Saki et al. [295] detail more drawbacks of quantum computing in general that directly affect QML. All of these directly or indirectly affect the security of QML: Any shortcoming or limitation might imply modifications, simplifications, and adaptions of the model. These in turn might have ramifications of lowered security. Options for adding security functionality (as well as any other functionality) are limited or removed. Generally, the increase in capability introduced by bigger quantum computers are invested in increasing model size, QML capabilities, and performance, instead of the secondary objective of security (or indeed any other secondary objectives). Naturally, this has been an observation throughout the development of computing in general, and is by no means exclusive to quantum computing.

Limited native gates: Quantum programs can consist of arbitrary quantum gates in theory, but since the number and type of native gates of NISQ devices is limited, the availability of all these gates is a very theoretical assumption. Thus, at a certain point in quantum circuit design/implementation, they have to be translated into native gates dependent on the hardware. This can lead to a higher number of gates and longer circuit runtime, and therefore overall poorer circuit performance.

Coupling constraint: NISQ devices have a limited connectivity that prevents two-qubit gates between certain (most) qubits. In today’s devices, for instance, most qubits have only two or three “neighbours” that can be used for two-qubit gates like `CNOT`. An example is depicted in figure 10 in chapter 11.3. To satisfy this coupling constraint, a compiler must add additional `SWAP` operations, which increase the runtime of the quantum program as well as the number of gates.

Cloud-based access: Since quantum computers are usually accessed via cloud services, there are security concerns, such as the risk of an attacker assigning inferior or in some way compromised hardware to the user without their knowledge. In addition, intellectual property could be stolen by an attacker viewing the structure or output of the quantum program.

Problem encoding: The design of the quantum circuit encodes the problem itself in various ways: for example by using certain gates (like problem-dependent rotations), or via the layout of input qubits. In many cases, it is therefore possible to derive information describing the (perhaps sensitive) problem by analyzing the quantum circuit.

Need for untrusted compilers: The circuit depth and number of gates can be optimized by (oftentimes untrusted) third-party compilers. The danger here is that the provider of the compiler may be able to derive sensitive aspects of the quantum circuit. Taking this thought a step further, the provider of the compiler might modify (or merely add to) the functionality of the resulting quantum circuit to his malicious needs.

Classical components: QML systems in production environments will continue to have a classical component for some time to come. That means that an attacker is not restricted to only the quantum components for his attack. They may use either, or even both, the classical and quantum space for their attack.

9.3 The ML/QML Pipeline

On a conceptual level, the computation pipeline of a QML system does not differ substantially from the pipeline of classical ML which has been described in section 4.1. Hence, many research topics and findings can be transferred from ML to QML.

The ML/QML pipeline, as part of the overall software development lifecycle, deals specifically with the creation of the ML model. The five general stages of the pipeline are shown in figure 7.

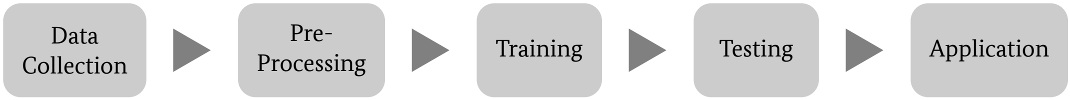


Figure 7: General Phases of the (Quantum) Machine Learning Pipeline

Papernot et al. [296] emphasize the need of recognizing that the broadening of ML deployments, as well as the expansion of ML techniques also gives rise to new vulnerabilities. They summarize current results about ML attacks, while also providing insights on defenses. This is done using an extensive threat model, which enables a categorization of both attacks and defenses. An important highlight is the tension—translating to a trade-off—between the complexity, accuracy, and resilience of ML models.

Gabor et al. [297] have a deeper look at the QML pipeline. Under the umbrella of acceleration of QML development, they point out that in addition to challenges inherited from ML, QML poses additional challenges. These include moving away from iterative training, and better integration and interaction of classical and quantum system components. The authors conclude in their paper that future research should deal more with the observed benefits of QML and their respective reasons.

In a previous study on “Vulnerabilities of Connectionist AI Applications: Evaluation and Defense” [298], security researchers at BSI provide a comprehensive view on threats as well as mitigation measures with regard to ML applications. Naturally, the application of machine learning using quantum computers is done to benefit from certain advantages over classical approaches, be it speed, robustness, or a combination of further advantages. Thus, in the context of QML, there can be significant differences with regards to threats and mitigation measures in each phase of the ML pipeline. These may depend on the purpose and the implementation of the used quantum algorithms.

> Machine learning algorithms enjoy a natural robustness against statistical noise and other imperfections contained in real-world data. On the other hand, the current generation of quantum computing devices suffers from various sources of noise. It is hence appealing to assume that QML methods require less error mitigation than their classical versions. Nevertheless, there currently exists neither a theoretical justification nor an empirical measure that allows us to quantify the inherent level of quantum noise robustness of QML methods.

9.4 The QML Pipeline in the Hybrid Setting

In a hybrid computing model, not all stages of a computational process have to be performed on quantum hardware. In most instances of the application of QML, either the training or the application phase will be carried out using a quantum computer. Other phases will remain in the classical domain for some time, most notably the data collection phase.

Data collection: Moving the data collection phase into the quantum domain will not be a widespread occurrence in the near future. A number of limiting factors are at play here: the amount of data involved, the different points in time of the collection and the processing of data, as well as the simple fact that most input data for a QML system will originate in the classical domain. This is true whether the data represents physical things and actions, or “purely” digital data is used.

Should these factors not apply, then even the data collection might happen within the quantum world. In this case, full advantages of a QML pipeline could be gained. An interesting application example could eventually be the use of a QML model to classify the results of quantum experiments, for example in the simulation of atoms or molecules using quantum computers. These kind of examples might prove to be the first end-to-end applications of quantum computing, where only the final output relies on classical hardware. For the area of quantum chemical research, further thoughts, estimations, as well as complications, are presented by Elfving, et al. in their review [299].

An interesting concept in the context of data collection is the notion of *synthetic data* which are especially used for increasing the amount of available training data or in cases of advanced requirements on privacy during the learning and application phases. To generate synthetic data, a ML system is trained on the existing training data. The ML system is then used to create further data that has the same properties (eg. statistical) as the underlying real data—it "looks" the same. Specifically for using a quantum annealer for creating a synthetic data set for cyber security, this idea is investigated by Dixit et al. [300]. For a more extensive consideration, also refer to chapter 18.3.

Pre-processing: In both classical ML and QML, this phase can be lengthy and complicated, or short and simple. Any means to clean up, transform, and potentially label the data points is carried out in this phase. The separation of data points into the training and testing sets is a requirement for any quality / performance indication during the testing phase.

Similar to data collection, this stage will mostly stay within the classical domain in the near future—in the case of the collection of quantum data, however, the pre-processing would benefit heavily from a quantum approach.

An important aspect is the encoding of classical data into the quantum world. As we have seen in sections 5 and 6, the (complexity and time) requirements of this task must not be neglected.

Feature extraction and feature selection can be considered part of this phase. Going one step further, the pre-processing can in turn be supported by classical ML or QML. The selection of meaningful features, for example, can be determined by a (sub-) classification step on a subset of the training data.

Training: Training an ML model is usually the most resource intensive stage of the pipeline. It is hence appealing to assume that this stage can benefit most from quantum computing and thus make a considerable difference. Existing quantum ML methods utilize the quantum processor as an inference machine: model predictions and gradient information is computed via quantum computation, while the actual learning step is conducted on a classical computer via classical numerical optimization methods.

The training procedure is carried out in an iterative manner. That is, training data is passed instance-wise to the quantum processor that runs a specific parametrized quantum gate circuit. From its output, the gradient of some loss function is computed. Here, the parameters for which we compute the partial derivates are precisely the parameters of the quantum circuit. A numerical optimization method, for example a plain gradient descent, is then applied to update the circuit parameters. This procedure is repeated until a convergence criterion is met. This could be, for example, the exhaustion of a pre-defined runtime budget.

In principle, one may unroll the iterations of the learning procedure into one large quantum circuit, avoiding the need for any classical computation. However, a quantum-classical learning pipeline in which only a single compute intensive sub-routine is offloaded to the quantum processor circumvents the limited number of qubits and the limited decohenrece time of todays' NISQ devices. On the other hand, considerable overhead is generated by input and output routines at the interface between the classical and quantum parts of the hybrid QML pipeline.

Clearly, the quantum computer cannot access data faster than the classical machine—given that the data resides in classical storage. Assuming the availability of some sort of quantum data storage happens frequently in the quantum machine learning literature. However, while research is being done in this direction, various limits exist—the most prominent example is the no-cloning theorem (see also chapter 3.4). Such assumptions therefore are mostly questionable, quantum storage of classical data cannot be realized by current generations of quantum computing devices. Under certain restrains, quantum storage ideas will materialize eventually. However, a “re-use” of quantum data for further computation at a later point in time will remain unlikely.

An interesting direction in the hybrid setting is the utilization of federated learning methods. One may circumvent limitations in terms of qubit count or qubit connectivity by fusing the resources of multiple NISQ devices, similar to the distributed learning setting (see also section 5.3.9). Further research on the possibility of overcoming certain limitations of NISQ devices using federated learning is needed.

Testing: The process of testing a computed ML model simply means applying it on the test data set (set aside during the pre-processing phase). Hence, testing should be performed in the same way the application of the model will be performed.

Generally, performing tests using classical hardware for applications carried out on quantum computers will certainly provide insights. For example, considering the noisy calculation and measurements of a (NISQ) quantum computer, advantages in the thorough examination of a computed model are obvious—many insights rely on the observation of the system during execution. This is much harder, for some observations inherently impossible, on quantum hardware. During a simulation, on the other hand, any and all changes during execution can easily be observed and analyzed. During the execution, individual states can even be modified easily, potentially leading to further understanding.

However, a comprehensive test can only be achieved in this case if testing is also carried out on quantum computers in this scenario. Likewise, where other stages of the ML pipeline have been carried out using quantum computers, and the application phase will be performed classically, then so should the testing.

Application: When moving the application phase of the ML pipeline into the quantum world, several disadvantages of quantum computing will have an effect. Apart from the obvious quantum encoding need for both model and data, the limited computation time on NISQ devices will again play an important role. This implies narrow constraints on the scenarios that will benefit from quantum advantages here. Most importantly, unless the QML model is described exclusively using quantum gates (avoiding usage of qubits for the QML model itself), a preparation of the quantum device, including encoding of the model data, is required prior to *each execution*. Input data, however, will always require state preparation prior to execution. To put it differently, we may formulate the question to what extent one can waive encoding the input data just because qubits have not yet been allocated in the definition of the model?

Nevertheless, depending on the (quantum-) generation or availability of the input data, as well as the (quantum-) processing of the output of the classifier, the advantages can easily outweigh the disadvantages. The example mentioned above, simulating atoms or molecules with a quantum computer, and then using QML as part of the processing of the results, might serve to demonstrate this notion.

Another often stated reason to consider quantum computing for the application phase is the hope for much better computation time given large or perhaps infeasible amounts of data compared to classical computation.

Reinforcement Learning: Another strategy for creation of an ML model is reinforcement learning. It can be based on both supervised and unsupervised learning, and should be considered a separate paradigm. Reinforcement learning exhibits different advantages and disadvantages, and is usually employed to solve different problems than the aforementioned approaches. It can generally be considered more advanced and also more complex.

Reinforcement learning is established in the quantum world under the term Quantum Reinforcement Learning (QRL) [301, 302].

9.5 Attack Surfaces in the QML Pipeline

As far as security is concerned, each stage of the pipeline presents its own attack surface both in the world of classical ML and in the world of QML. Wang et al., investigate the attack possibilities and vulnerabilities for each stage of the pipeline for classical ML [303].

With respect to potential attack surfaces on QML systems and solutions, we thus consider the ML/QML pipeline as discussed above, in addition to the transfer of insights from the classical- to the quantum world as depicted in figure 5. An overview of various attack methods on the individual phases of the pipeline is provided in figure 8. These attacks originate in the classical space, and are transferable to QML. In the following, we derive the methods of attack on QML, and explain these and other attack surfaces.

Due to our focus on security in quantum machine learning, the following discussion is kept short. It is mainly intended to support later chapters and the more detailed considerations of QML security discussed therein. Those kinds of attacks on classical ML which are (in theory or practice) not transferable to QML, will therefore not be mentioned or it will only briefly be explained and justified why they are not relevant for the context considered here.

In order to identify potential attack points and attack types on QML, we first list considerations as to classical ML and how these may transfer to QML processes. For instance, it is known that, in the world of classical ML, data collection and training processes are vulnerable to attacks. We thus have to investigate if and where related points of attack can be identified with regard to training QML systems and what defense mechanisms, if any, could possibly be derived. Indeed, given the stages of a classical ML pipeline as illustrated in figure 8, one can immediately conceive of various attack points on QML. At this point, it is worth distinguishing the term stealthy channel from the term side channel (used later in Section 11.5). Stealthy channel means a gateway for the ingestion of fraudulent (poisoned) data.

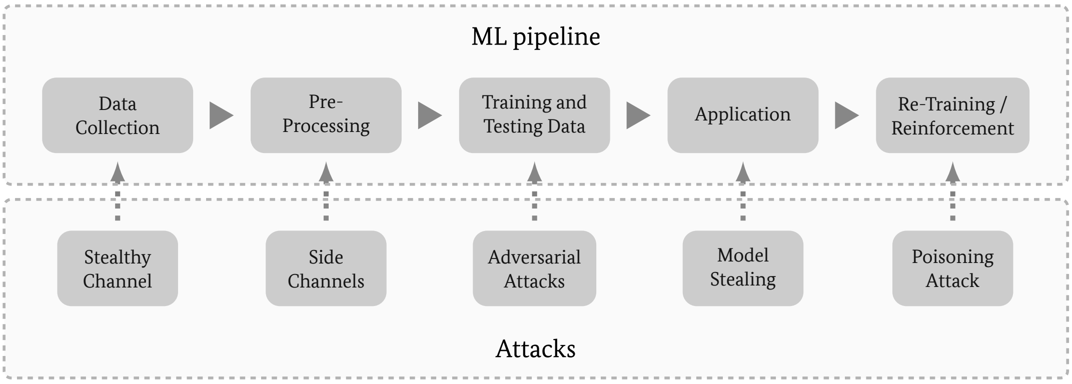


Figure 8: Attack examples in the ML pipeline

Data collection happens before any model can be trained. This collection of relevant and representative data requires care because, if data is manipulated or poisoned at this early step of QML system development, each subsequent step would obviously suffer. Accidental shortcomings in this phase can have deep implications for later development phases and especially for the fully-trained model at the end of the whole process.

From the perspective of security, additional care has to be taken. An attacker might be able to add data points, remove them, or change certain aspects of them. This might even happen accidentally—not all threats to the model require an intentional attack, as, for example, the availability of the machine learning application (for example within satisfactory boundaries of performance and accuracy) is fragile. Even minimal changes to a small number of data points, or even the change of a single data point, can have disastrous results for the final ML model. In addition, even with rigorous data cleaning and quality maintenance processes, such changes might go unnoticed. This concept of “garbage in, garbage out” obviously applies to QML in the same way as for any data processing system.

Training and testing data form the basis for a well-functioning ML model. The accuracy of the model depends significantly on the correctness and comprehensiveness of the data on which it is trained. If these are compromised, the result may be an inaccurate and therefore unusable model—or worse. The same vulnerability exists in QML systems: if a QML model is trained based on malicious data, almost arbitrary behavior can be the result. Nevertheless, QML methods might be subject to a high level of noise during training, which can make them more robust to small changes in the data itself.

Feature extraction refers to the process of deriving features from the original data that preserve the information and are not redundant. It can be attacked by defining wrong features or limiting the feature set. A QML system has the same vulnerability to such an attack, since using wrong or incomplete features decreases the quality of the final model in the QML space as well.

Learning is the phase in which the algorithm is trained to create the right output by modifying the model parameters accordingly. In this phase, the parameters of the algorithm could be compromised and thereby the performance of the global model. Also, the resources used during learning could be influenced, which might give an attacker a (slightly less directed or exact) method of modifying the resulting model.

If an attacker has access to the computing resources during the learning phase, they may gain knowledge of the weights (and, by extension, the gradients) of the model. They can use these for finding adversarial perturbations which maximise the model’s loss on a specific input. While this kind of attack does not require an attack surface specifically during the learning phase (the knowledge about learned information can be gained at a later stage with the same results), the attack implementation is more stealthy and it is more difficult to discover adversarial samples than it is the case of attack implementations relying on access to the classifier, instead of the weights and gradients.

Application happens after the training of the model is completed. At this stage, many different types of attacks are possible, depending on the knowledge the attacker has and how the system is used. Malicious input (data that has been altered, perhaps imperceptibly, and appears legitimate) could be used to fool the model and provoke false classification. In addition, brute force attacks could be used to reconstruct the model, or to extract potentially confidential data that was used to train the model. As Lu et al. [304] point out, black box attacks are transferable from classical ML to the quantum world in general. Also, attacks on the implementation itself, known as side-channel attacks, are possible. Quantum systems have a much smaller attack surface to side-channel attacks: for example, classically relevant methods like power analysis or timing attacks based on duration of loops or branches, are not applicable to quantum systems—however, they might be part of hybrid quantum-classical systems. On the other hand, side-channels such as maliciously added qubits, coupled with entangling circuits, can leak (or rather deliver) data to an attacker with respective capabilities.

Re-Training of the model occurs in certain, reasonably defined time intervals, depending on the application. An underlying assumption is that any future data is similar (within some boundaries) to the training data. However, if the properties like the distribution of the data change significantly, the model must be adjusted to take this into account. For this purpose, the entire existing pipeline must be run through with new data. This results in the same weaknesses as mentioned in the earlier stages.

Reinforcement is typically used to enable a model to continuously improve itself. In contrast to supervised and unsupervised learning, the model is not only based on an initial (potentially labelled) data set, but learns from perception and interpretation of its environment. It can be considered a separate paradigm and approach, and thus many ML models do not make use of reinforcement. The long-term goal is achieved through trial and error, rewarding desired behavior and punishing undesired behavior. These models are therefore particularly vulnerable to malicious input that may appear legitimate and thus lead to unwanted learning behavior. Since adversarial reinforcement learning approaches exist in the classical world as shown in [305], similar threats to QRL exist.

9.6 Utilization of Quantum Computing for the Training Phase

From the point of view of present day ML, benefits of quantum computing are mainly anticipated for the training phase of a learning system. This is because present day ML deals with complex models with many degrees of freedom whose training is a very time consuming endeavor. Consider, for instance, OpenAI’s GPT-3, a large transformer model for text analysis and synthesis that achieves human-like capabilities [306]. According to its developers, the training of this system took dozens of petaflops per second days[3], utilizing a large-scale, high performance GPU cluster. With respect to a single, high-end GPU, this translates to several hundred years of compute time. Against this backdrop, there are expectations that quantum advantages may considerably accelerate training once quantum computers reach their full potential. On the other hand, when running a trained system in practice, its computations on a given input typically only require fractions of a second. In this sense, quantum-training seems considerably more relevant than quantum-inference.

Saki et al. [295] mention a bottleneck of QML, namely that during training in parameterized quantum circuits, gradients can vanish. From these vanishing gradients, also called *barren plateaus*, optimization methods are not able to train the network further—no escape from the plateau is found. The authors also mention that quantum noise can cause this ef-

[3] A “petaflop per second day” consists of performing 10^{15} neural net operations per second for one day, or a total of about 10^{20} operations [307].

fect. McClean et al. [308] point out that quantum neural networks suffer from the effect of barren plateaus, and that with an increase in the number of qubits, the speed of gradients approaching zero increases exponentially. Thus, this training problem, if it remains unmitigated, places a severe limit on the capabilities of QML. Abbas et al. [309] use the tool of the "Fisher information spectrum" and apply it to barren plateaus. They discuss that QML might even have advantages over classical ML with regards to this problem.

The issue of barren plateaus are not specific to QML, they are extensively studied as a part of classical ML as well. In fact, barren plateaus are a major issue in the training of neural networks. Thus, at the current stage of QML research, it is expected that both advantages and disadvantages of quantum models are discussed and challenged extensively. Research of the phenomenon itself, as well as measures to counter the effects, might lead to significant improvements in the field of neural networks.

10 Classification Dimensions for Attack Types

Categorization and classification aids in the examination of characteristics, differences and similarities of the different attack approaches both for ML and QML. This classification can be done among multiple dimensions. From a general point of view, we will consider three fundamental criteria explained in the sections below. These have been introduced by Barreno at al. in [310] for the purpose of attack analysis for classical ML.

10.1 Influence Capabilities of the Attacker

The first criterion gauges whether the attacker is able to actively manipulate, or passively observe. Naturally, this is not a binary choice—the potential for manipulation can be broad or narrow. It can even be connected to an increase in detection risk (from the point of view of the attacker) and therefore a trade-off between risky but extensive manipulation on one hand, and comparatively safe but limited manipulation on the other.

Similarly, the ability to observe can be considered broad or narrow. More importantly, either ability must be regarded separately: the ability to manipulate does not imply the ability to observe, and vice versa.

In addition to that, either capability can be narrowed down to specific phases of the ML or QML pipeline. As an example, manipulation can occur at the data collection stage only, or observation might only include the learning, but not application phase.

10.2 Attacked Property of the System

This dimension pertains to the principal security goals commonly considered in IT security: *integrity*, *availability*, and *confidentiality*.

Broadly speaking, an attack on the confidentiality of a ML/QML system is connected with the objective of extracting some piece of information that the defender does not intend to disclose. An important category are *privacy attacks*: here, the attacker extracts information about the training data set. Similarly, *model reconstruction attacks* aim to extract information about the ML/QML model used—usually as a means of reverse engineering the whole model. This can be a preparation of further attacks.

Attacking the integrity of the model is achieved when the model misclassifies input data based on the actions of the attacker. Depending on the purpose and implementation of the model, it might either be a binary classifier that places each sample in one of exactly two classes, or it might be a more sophisticated classifier that provides probabilities for the sample belonging to each of any number of classes. A misclassification thus can either mean that a sample is not placed into the correct class, or that the sample is placed into a specific incorrect class. Additionally, the aim might be narrow: a few selected samples, or even only one specific sample, shall be misclassified by the model. Conversely, the attacker might target for a whole range or category of samples to be misclassified.

Lastly, the availability of the model is attacked when the ML/QML system becomes unusable. This means that a very large number of inputs are misclassified. In a binary classifier, this includes both false positives and false negatives. In a classification system with multiple output classes, this might affect the output probabilities for any subset.

10.3 Attack Specificity

The specificity of an attack is a measure of its precision. It distinguishes an attack that is extremely narrow and targeted from one that is very broad and non-specific.

For example, the misclassification of one particular input into one chosen output class is the goal of an attack with high specificity, while in an attack with low specificity, the attacker might have more flexibility and therefore might only want to aim for the misclassification of *any* (or any number of) inputs. The concept not only applies to misclassification, of course, similar examples can be found for other attack aspects and dimensions. For instance, in privacy attacks the range might include identifying one particular training sample versus gaining the knowledge that at least one sample out of a set has been used during the training. In an attack that involves manipulating or observing training data, the attacker correspondingly can have highly specific or non-specific goals.

For the current NISQ devices, a level of specificity arises with respect to knowledge about the underlying quantum hardware. Here, an attack might be successful in case the attacker has knowledge about the specific device used, and specific properties like individual qubit error rates, gate fidelities, or decoherence times. Certain side-channel attacks are based on this knowledge, for example. While the acquisition of this information certainly represents a unique challenge for attackers, we will see that the exploitation of this information can open up opportunities for novel attacks that have no classical counterpart.

10.4 Attack Types

The literature presents different ways how attack types can be differentiated. Kumar et al. [311] deem it necessary to understand the reason for a ML system's failure. They differentiate between two categories of causes, firstly the intrusion of an attacker, and secondly the inherent design of a system. The authors name a few other weaknesses and attack points of a classical ML system such as exploiting the system's software dependencies (e.g. using buffer overflow), attacking the ML supply chain (in their example, checking in malicious/poisoning code into the framework that hosts image recognition models).

To categorize and assess attacks, Berghoff, Neu and von Twickel [312] address IT security of ML applications with respect to threats to integrity, which are significant for ML-based computer vision applications, for example. A holistic view of integrity requires considering interpretability, robustness, and documentation. The authors provide a broad list of both threats and feasible remedies based on the current state of the literature, and they discuss adversarial attacks and ML-specific vulnerabilities including respective ML-specific reasons. For this purpose, The authors follow the ML lifecycle from the planning, data acquisition, training and evaluation phases up to the application. They list the goal, knowledge of the attacker, efficiency and the availability of mitigations as criteria for categorizing attacks.

Novel attack types arise from the new specificity types mentioned above. The level of processor instance-based low-level hardware information requires additional measures for securing the top-level application. Here, one has to distinguish cloud-based quantum hardware and dedicated quantum hardware scenarios: Cloud-based quantum services usually offer a wide range of processor specific attributes to ease the engineering and optimization of the quantum-gate circuits. Thus, knowledge for an attack can be acquired from the cloud service or its API. For instance, quantum computation frameworks provide routines that give access to attributes such as `processor_type`. On the other hand, when we assume that the defender utilizes their own dedicated quantum hardware, such low-level hardware-based attacks become much harder. Knowledge about the mere processor type is insufficient and this kind of information can only be obtained at the cost of a large computational overhead on the software side. In other words, using dedicated NISQ quantum hardware in a production environment can increase the actual security. One can, however, speculate, that a malicious attacker is willing to exhaustively probe noise patterns in the output of a quantum computer to match them against known characteristics, and thus to determine the kind of device they are dealing with. Such procedures will still work in the post-NISQ era when quantum computers of considerably reduced imperfections can be produced at scale. This is because the quantum system in the processor will always have to have some form of interaction with its environment—at least via the classical logic required for data in- and output. In other words, it can be expected that quantum and classical data processing systems will remain susceptible to this kind of probing.

In academia as well as in industry, the categorization based on the knowledge of the attacker is prevalent and will be used in the following. We note that an attacker can have different levels of knowledge about the system they want to attack [313]. With **perfect knowledge**, the attacker is aware of all essential aspects of the ML system to be attacked; such perfect knowledge could be so specific as to include, say, all the details about the weight- and bias values in a neuronal network that is to be assailed. In case the attacker has perfect knowledge, the attack is denoted as a white box attack.

As attackers have perfect knowledge of the target model in a white-box setting, they can design adversarial examples that are misclassified during test time (evasion attacks), and inject carefully designed examples at time of training to confuse the classifier and thereby reduce its accuracy (poisoning attacks) [303]. Many models are open source (for example ResNet [314]) and therefore it is not unreasonable to assume that attackers know about the model nature to perform whitebox attacks.

Usually, the major purpose of ML is to processes large amounts of data. Take, for example, a computer vision scenario for an automated car—vast amounts of input data from various cameras and other sensors like radar and lidar needs to be processed with low latency. Thus, computing in the cloud is often not an option. Therefore, by design, the ML model and all processing is done within the (often publicly available) device or product. In turn, this means that a determined attacker can be assumed to possess everything required for an attack: while defense mechanisms might offer slow-downs and complexity increases, with the device in the hands of the attacker, the extraction of any required information is fundamentally possible.

Furthermore, other devices of the same kind or manufacturer are likely to contain a similar or even the same ML model and application software. Thus, after extracting the information from one device, and subsequently developing an attack, all such devices can be attacked.

Another instance of white box attacks is the so-called *disgruntled employee scenario*: an employee might have limited or even full access to the ML model, its input data, and even to the ML training process itself. (This is where classical operational security can offer some security.)

For QML, perfect knowledge could include information about the specific hardware or implementation details like circuit parameters and angles of rotation gates. Both specify the QML model itself, for example a neural network topology and its weights, or the branches of a decision tree. Moreover, hardware information entails the qubit connectivity, CNOT error rates as well as decoherence times. For adiabatic quantum computers, the software is fully described by the problem (target) Hamiltonian and the annealing schedule. On the hardware side, it includes information about the qubit connectivity, and integrated control errors.

The mapping of logical or algorithmic qubits to physical qubits can play an important role: given perfect information, an attacker could try to manipulate the input data in a way that involves qubits with relatively high error rates in the classification. The desired result might be an increased chance for a misclassification.

Especially with the usage of quantum cloud services and the full publication of information about these hardware properties, the feasibility and potential impact of this kind of attack deserves some research attention.

Second, with **limited knowledge**, the attacker has some amount of knowledge about the ML system that is to be attacked. This knowledge might include data about the type or structure of the classifier, or about the data set and features with which the system has been trained. Of course, the spectrum is fairly wide—some attacks require extensive prior knowledge about the ML system in question (leaning more towards perfect knowledge), others might require small pieces of information only (leaning towards zero knowledge).

In case the attacker has limited knowledge, the attack is denoted as a grey box attack. A grey-box attack conceptually lies between white- and black-box attacks since the attacker does not possess all, but some knowledge about the underlying model structure. Common examples of the kind of knowledge an attacker has for grey-box attacks include feature representations and learning algorithms [313].

Consequently, with **zero knowledge**, the attacker has no prior knowledge about the system and its properties. For the attack, they can utilize only the system itself in its current method of operation. In the usual model, this includes any number of classification input and output pairs. However, in practical implementations, additional barriers might be added as supplementary defensive measures even against an attacker without prior knowledge—for example rate limiting, or other limits regarding the kind of input that is accepted by the system. On top of that, the actual output of the system might not be directly available to the attacker.

With zero knowledge, the attack is denoted as a black box attack. Wang et al. [303] mention the adversarial capability as a criterion for delimiting black box attacks from white box attacks against classical ML systems. In the course of a black box attack, an attacker deduces the ML system's vulnerabilities by observing the model's behavior during prediction or test phase.

11 Attacks on Quantum Machine Learning Systems

Quantum Computing in general, and consequently QML, is not immune to attacks. In their paper entitled *A Survey and Tutorial on Security and Resilience of Quantum Computing* Saki et al. [295] give an outline of several security threats, attack methods, and countermeasures. The authors point to the fact that quantum devices are vulnerable to threats "from insider and outsider adversaries including input tampering, program misallocation, fault injection, reverse engineering and cloning". Saki et al. review as well the relationship between resilience and security. In the following, we apply a similar methodology using these four types of attack to the specific case of QML systems. What the authors call *input tampering* is synonymous with the term evasion attack used by us (and also widely in the literature). Analogously, the term *fault injection* corresponds to data poisoning, *reverse engineering* to privacy attacks and *cloning* to model stealing in our nomenclature. With this we have already mentioned the most important attack methods, which are described in the following subsections.

11.1 Model Stealing Attacks

Model stealing refers to an approach in which an attacker rebuilds the model using carefully designed queries. This procedure is probably just as dangerous for QML as for classic ML because an attacker can use the stolen model (which is thus fully available to him) to derive further knowledge regarding tricking or evading the QML-based application—be it a spam filter or an intrusion prevention system. Notably, losing the model to an attacker means a severe loss of intellectual property. As with privacy attacks, in the case of current NISQ hardware, it should be noted that the inherent noise of the hardware is likely to prevent an exact reconstruction of the model. Due to the properties of QML systems, certain physical impossibilities hinder an exact reconstruction. However, a sufficiently exact reconstruction might be within the realm of possibility. As efforts are made to minimize noise over the long term, model stealing is relevant to QML models.

Rigaki and Garcia [315] distinguish between attacks against centralized learning and attacks against distributed learning. They consider *shadow training* as an established technique that is widely used in the field of supervised learning. Here, an attacker mimics the behavior of the target model by training a set of shadow models on shadow data sets that are usually assumed to possess the same distribution as the target data set. The terms "meta-model" and "surrogate model" are often used synonymously and several so-called shadow models are used to obtain a meta-model of the target model. This fashion of stealing a model is a means to the end of another type of attack—the membership inference (a privacy attack)—which we will discuss in more detail later in section 11.4. The output of these shadow models, including the known labels of the shadow data sets, will be used for training a meta-model that the attacker utilizes for the membership inference. Similar kinds of attacks can, in principle, also be imagined for QML as the basic idea behind this procedure appears to be platform agnostic.

Although it is not a true model stealing attack in the strict sense, the following is a related and dangerous attack that uses an existing model, including its API, to generate samples that the model classifies positively. A dangerous use case is the generation of synthetic biometric data, such as a synthetic fingerprint, based on which an authorization system allows access to a security-critical system. Poh [316] demonstrates the vulnerability of a fingerprint recognition system by combining two model stealing techniques, the wolf attack and the hill climbing attack. His starting point of model stealing is to assume that the attacker has access to the Software Development Kit (SDK), and iteratively applies the SDK according to the pot-beating principle. Hill climbing iteratively reconstructs a fingerprint from an unrelated sample by presenting modified variants. By improving the result step by step, the attacker undermines the system's security by learning its strengths and weaknesses. The wolf attacks, on the other hand, does not synthesize new samples, but starts with a large database of biometric samples and tries to find the best match of a target fingerprint to be reconstructed.

Attacks like these can also be imagined in the context of QML systems. However, due to inherent quantum computing characteristics such as measurement noise, it is clear that significantly more samples would have to be drawn for these kinds of model stealing attack to work. In this sense, QML systems still seem to be inherently more secure against these kinds of attacks. Furthermore, both kinds of attack depend on the attacker's level of access to the developed model. Given the present state of quantum computing technology, where access to quantum computers is offered via web interfaces and APIs, this may or may not be the strength of QML systems. On one hand, access to quantum computers is inherently guarded; on the other hand, malicious attackers may devise tools specifically targeted at monitoring the traffic to and from quantum computing interfaces.

Further model stealing considerations are presented by Ratke [317]. The author is concerned with adiabatic quantum computing (AQC) (see section 3.1) for quadratic unconstrained binary optimization problems (QUBOs) which are of considerable interest for a wide range of practical and industrially relevant settings (for example for budget allocation, RNA folding, or routing). Since access to adiabatic quantum computers is provided by only a few companies and takes place via web interfaces where users upload the parameters of the particular problem to be solved to obtain solutions, he raises two critical questions:

- can providers reverse engineer QUBO parameters to infer what kind of problem they encode and thus glean insights into business relevant information?
- given that QUBOs often consist of linear combinations of individually weighted QUBOs, can providers infer the corresponding weights which again may reveal potential business secrets?

Ratke approaches both questions by training conventional ML models to predict either QUBO matrices or penalty coefficients. His (preliminary) results suggest that it is, to some extent, possible to infer what kind of problem a QUBO matrix encodes as well as how supposedly known constraints have been weighted in a specific problem instance. Ratke's overall approach thus points to a potential weakness of present day AQC platforms and indicates that QUBOs are, in principle, reversible. It therefore seems plausible that a malicious agent with access to considerable resources (high performance classical hardware) might be able to infer business details from analyzing API traffic of web-based AQC services.

> Indeed, this hypothesis seems testable. Future research projects could systematically explore to what extent the inputs to an adiabatic quantum computer allow for deducing what kind of practical problem they encode and how this problem has been modeled in detail. Given that adiabatic quantum computing may play a role in sensitive industries such as the financial services industry [318], such research seems warranted.

Clearly, QAOA-based quantum-gate methods are QUBO solvers and hence suffer from the very same vulnerability. Analyzing the API traffic can reveal the underlying QUBO matrix Q and hence private information. Even when the problem is transmitted in form of a Hamiltonian operator H, the sparse underlying data format allows the recovery of the QUBO coefficients up to a constant c. More precisely, the QUBO problem can be equivalently encoded as $H + cI$, where c is some unknown constant, and I is the 2^n-dimensional identity matrix. In current implementations, only H is required to run QAOA. It depends on the problem at hand whether the missing knowledge about the constant c can help to reduce the security risk. If this is the case, a similar transformation can in turn be applied to the QUBO matrix itself, which might mitigate the shortcomings of AQC.

It is conceivable that this reasoning applies to variational quantum circuits as well. However, the function space that is represented by general quantum circuits is even larger than the space of QUBOs. Thus, recovering the probabilistic function implemented by the circuit requires a full state vector simulation, which can be considered as intractable. One has to recall that this problem is very hard, even in the case of digital circuits.

11.2 Adversarial Attacks

There are a variety of adversarial attacks that have been developed and improved over the years. Figure 9 shows a timeline from 2014 to 2021. A comprehensive review of adversarial deep learning has been carried out in [319]. The classification of attacks (evasion, poisoning, backdoor, model and data extraction) mostly coincides with ours, and similarly, the authors orient themselves according to the points of attack in the life cycle of an ML system.

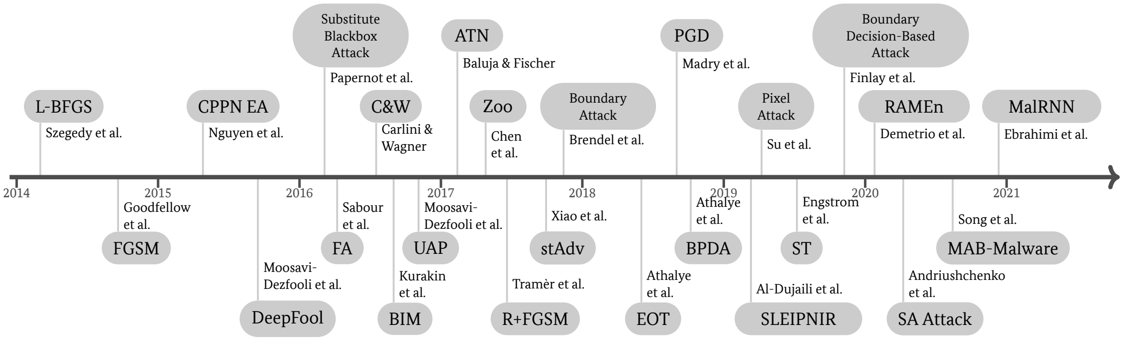


Figure 9: Overview of different adversarial attack methodologies

Biggio and Roli [313] emphasize how safely and reliably an attacker can undermine the predictions of high performing classifiers via adversarial input perturbations that are applied during training or at the test phase. With respect to computer vision and security, the authors give an overall view of ML's vulnerability to adversarial examples (so-called "wild patterns"), including corresponding countermeasures that have evolved over the last decade. The authors consider the upcoming stakes in developing secure ML algorithms, which forms a basis for our implications on QML.

A well-known problem of ML models, such as neural networks, is that they are prone to adversarial examples. These are inputs that are intentionally perturbed reaching a worst-case (adversarial perturbation) in such a way that they are highly likely to be misclassified. Goodfellow et al. [320] argue that the cause of this phenomenon is not, as is often assumed, the strong non-linearity of the model, but is actually a result of overly linear models and a property of high-dimensional dot products. This view allows the authors to present a method for quickly generating adversarial examples for adversarial training, in which adversarial examples are included in the training process to make the model more robust against adversarial attacks. This training can also lead to regularization.

Since QML takes the approach of mapping ML directly to the quantum world, we assume that it has the same weaknesses as ML. Specifically, if we follow the author's argument that the vulnerability to adversarial examples is a consequence of overly linear models, QML has the same weakness since it consists mainly of linear circuits and uses linear algebra. Because the structure of quantum computation is specified by a product of various unitary matrices, errors that happen early in the quantum circuit will propagate and amplify with increasing depth.

Thus, one can observe that linear models that are easy to optimize are also easy to perturb. At the same time, models that are supposed to be resistant to negative perturbations require a structure with additional hidden layers and are therefore difficult to optimize. Goodfellow et al. are also able to give a rationale for an interesting property of generalization of adversarial examples, namely the fact that an example generated for a particular model will also be misclassified by other models with different architectures and training data. This can be explained by the fact that the adversarial perturbations show a significant correlation with the weight vectors of a model, and several models trained for the same task learn similar functions. The weight vector constitutes a generalization of the classifier. The attackability thus also applies to similar models. It should be noted that the authors' statement is limited to specific examples.

If we accept the idea that chained matrix dot products are responsible for error propagation in classical ML, especially in classical neural networks, then QML error propagation has to be limited. This is because matrix operations in QML (except for measurements) can, by the very nature of quantum mechanics, only involve unitary matrices and are therefore norm-preserving. This observation is independent of the kind of data (adversarial, or poisoned via error injection, or other) but applies to the processing of any quantum encoded inputs.

> Compared to error propagation in classical ML algorithms, error propagation in QML algorithms raises research questions that still need to be investigated more deeply. This is because error propagation in QML is limited due to norm preserving properties of the unitary operators acting on quantum states.
>
> Whether or not this leads to increased or decreased robustness of a QML procedure compared to its classical counterpart is a question that still awaits a definitive answer and should be investigated in separate research.

It is well known that certain data domains are more amenable to adversarial attacks than others. For instance, a study regarding medical imaging [321], found that highly structured input data leads to more high-gradient regions, which, in turn, are susceptible to adversarial perturbations. Moreover, the authors claim that expressability of state of the art DNNs achieved through vast over-parametrization is an important weak point for exploitation by adversarial attacks. Parameters which are virtually unused offer additional degrees of freedom when seeking adversarial examples. Due to quantum entanglement and the specific parameters of quantum circuits, measuring the expressability of variational circuits is more subtle than considering the sheer number of parameters. Various measures for expressability and entanglement capability have thus been identified [322]. However, it is an open question whether these measures can be connected to a predisposition for adversarial attacks when considering variational quantum classifiers.

> QML often relies on generic variational (parameterized) quantum circuits. Expressability as well as the entanglement capability of such circuits decide on the quality of the learning result. However, both properties are unknown beforehand.
>
> Investigating new methods for predicting expressability and entanglement capability will facilitate the assessment and comparison of different QML models. Furthermore, they might prove to be useful indicators of the susceptibility of a QML model for adversarial attacks.

Aldahdooh et al. [323] consider current detection methods for evasion attacks (attacks where a network is fed an adversarial example) for neural networks performing image classification tasks. They provide a categorization of these detection methods according to the attacker's level of knowledge (white-, grey- or black-box) and the technique used (supervised or unsupervised), and perform an experimental study on four publicly available data sets using selected detection methods. They find that the tested detectors lack generalization, i.e. the ability to detect the wide range of white-, black- and grey-box attacks, as well as counter-counter attacks. They take this as an indication that more research should be done in this area. So called adaptive attacks should be mentioned as well—adaptive attacks were specifically designed to target a given defense, but are not the focus. Counter-counter attacks are attacks on countermeasure methods.

Adversarial attacks are already being investigated and discussed in the field of QML as well. For instance, Liao et al. [324] observe a vulnerability of QML models for classifying so called Haar-random pure states which are quantum states resulting from applying a randomly chosen unitary operator where the choice happens with regards to the Haar measure on the group of operators. To be specific, they observe a vulnerability to small, targeted perturbations which is more severe than the well known perturbation vulnerabilities of classical state of the art neural networks. The authors attribute this effect to the so called concentration of measure phenomenon, a characteristic of the metric space in probabilistic sampling. By investigating the robustness of their quantum classifiers for actual cases, the authors find that with increasing qubit number, the vulnerability of classifying increases. Concretely, the vulnerability of classifying encoded states that have been sampled from a Gaussian latent space is much weaker (robustness decreases mildly polynomially) than the vulnerability of classifying Haar-random pure states (robustness decreases exponentially). While the question investigated by the authors is of rather academic nature and of limited practical concern, it nevertheless establishes that the quantum nature of QML solutions does not make them inherently more secure than classical solutions but may, at least in certain cases, also be a drawback.

From the point of view of Liu and Witteck [325], high-dimensional classification tasks are a strength of QML. However, precisely in this regard QML seems to be more vulnerable to adversarial attacks, regardless of the classification model. The authors demonstrate that the amount of perturbation an attacker needs in order to cause a misclassification scales inversely with dimensionality. They note that a compromise must be found between an algorithm's security against such attacks and its quantum advantage for solving high-dimensional classification problems.

Lu, Duan and Deng [304], too, investigate the potential for QML attacks and conclude that QML systems, similar to conventional NN-based classifiers, are vulnerable to adversarial examples as well, regardless of whether the input data is quantum-based or classical. The authors observe that quantum classifiers reaching very high precision are clearly susceptible to adversarial examples.

Further, they concretely illustrate this vulnerability of QML methods in several settings, for instance in the classification of images (images of handwritten digits—the MNIST data set[4]), in the learning of matter phases (ferromagnetic orders and symmetry-protected topological phases), and in quantum data classification. In addition, the authors demonstrate that by drawing on the information in these examples, defense mechanisms can be built to counter such attacks. One approach which the authors highlight is so called *quantum adversarial training*, a strategy that pursues the increase of model robustness through augmenting the training set by adversarial examples. By implementing a *robust optimization*, the authors demonstrate how to improve this strategy using a clever generation of adversarial examples to be included in the training data set.

Gong and Deng [326] investigate the vulnerability of QML system to adversarial examples and perturbations and their universality. The authors show empirically that there exist *universal adversarial examples* which are able to trick several quantum classifiers. Moreover, they prove the theorem that a small increase in the perturbation strength of $\mathcal{O}(\frac{\ln k}{2^n})$ is sufficient to represent a moderate universal risk for k classifiers that receive n qubits as input data. In their study, each classifier has its own hypothesis function and is independent of the others—no combinations are considered.

> There is a high research interest in adversarial attacks against both ML and QML systems. Overall, the examination of an inherently improved defense of a QML system, especially against adversarial attacks, might have highly valuable results. Conversely, specific added countermeasures are needed to oppose the attacker.

11.3 Data Poisoning Attacks

Attacks based on data poisoning are firstly already very common in scholarly ML considerations and secondly relevant for QML, since QML only works effectively and correctly if—analogous to classical ML—the (quantum) data quality is sufficiently high. In addition, encoding the training data for the quantum learning process is a separate, possibly

[4] `http://yann.lecun.com/exdb/mnist/`

resource-intensive process step. Therefore, QML is vulnerable to poisoning attacks as it is the case for classical ML. Additionally, due to the data fuzziness in the quantum environment, the quality requirement is higher, which would imply a higher data sensitivity. Conversely, fuzziness (and its dealing with) in the quantum domain could lead to increased robustness. Since quantum computing hardware suffers from a vast amount of different sources of noise, effective ML methods must already exhibit noise robustness to some non-trivial extend. Thus, for a successful attack to have an effect, the poisoning amount must be higher than the regular noise floor of the quantum computing device. Quantifying what strength of a data poisoning attack is required to supersede the regular noise floor is an open research question.

> Quantum algorithms exhibit an inherent robustness against noise. Clearly, QML methods must share these properties, which leads to the conclusion that QML methods are not as susceptible to noise as their classical counterparts. One may hence conjecture that an attacker requires stronger data poisoning attacks in order to break QML systems. It is an open question whether this difference is significant or even model specific.

There are two major types of poisoning attacks, namely those that are aimed against the ML system's availability and those that attack its integrity [327]. In the first case, the attacker attempts to inject so much bad data into the system that the model becomes effectively unusable. In the second case, or so called *backdoor attacks*, attacks are more sophisticated and aim to make the classifier work exactly as it should, with a single exception: a backdoor which the attacker uses to make the ML system perform whatever he desires [327].

In a white-box scenario, low-level properties of quantum processors can be exploited to realize availability attacks as follows: Each qubit and each connection between qubits in real-world quantum computers are subject to different amounts of noise—a visualization is presented in figure 10.

Clearly, when qubits and couplers with high error rates are involved in a quantum computation, the final outcome of the computation will suffer from higher noise than expected. In variational quantum classifiers or quantum support vector machines, specific qubits and qubit connections are responsible for processing specific data features. The assignment between qubits and data features is fixed during learning and cannot be easily changed during runtime, since the learned model parameters are also subject to the specific noise signature of the selected qubits. In practice, this effect can cancel out on average, since feature values can be distributed evenly over low error and high error components of a quantum processor. However, when an attacker has the opportunity to manipulate data points such that high error components are utilized more frequently, the computation can be distorted up to some unforeseeable extend. Thus, quantum computing resources might be more amenable to availability attack—at least in a white-box setting. On the other hand,

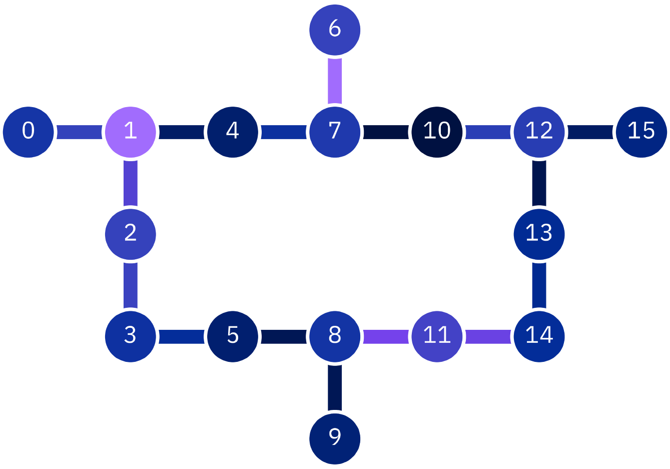


Figure 10: Examples of errors observed in practical experiments on an IBM Quantum System. The shading of the qubits 0–15 indicates the readout error rate, which is measured between 1.0% and 3.2% (qubit 1). The shading of the connections indicates the CNOT error rate, which is measured between 0.6% and 2.3% (connection of qubits 6 and 7) [328].

targeted attacks (backdoor attacks) are likely to be more difficult due to the general high noise floor.

Gu, Liu, Dolan-Gavitt and Garg [329] argue that outsourcing the (expensive and time consuming) training process of Deep Learning Nets opens a door for security breaches such as building maliciously trained or backdoored nets possessing a high performance while behaving badly, behaving in the attacker's spirit upon certain inputs. The authors illustrate the effect of backdoors under a realistic scenario by building a classifier which confuses a stop sign with a speed limit sign if a special sticker is attached to the stop sign. They also demonstrate that this backdoor persists even after retraining the model.

In their detailed and much noted recent paper, Goldwasser, Kim, Vaikuntanathan, and Zamir show how a backdoor can be introduced into any model [330]. They specifically look at the realistic scenario where the learning stage is done by a service provider and carefully construct several undetectable backdoors which can be used to maliciously change the output behavior of the trained classifier. One such backdoor utilize digital signatures schemes where input to the classifier is considered as message-signature pair and the classifier system trained by the malicious service provider is augmented by a public-key verification procedure. This procedure runs in parallel to the actual classifier, gets triggered by malicious message-signature pairs, and, when activated, bypasses the classification mechanism to produce output unintended by the customer. Clever minuscule changes of an input can thus be made in a manner such that, from the point of view of the unsuspecting customer, they are virtually indistinguishable from normal input but produce message-signature pairs which lead to whatever output behavior the rouge provider wants. Moreover, when certain specific training methods are used (Random Fourier Features / RFF, or Random ReLU networks), this mechanism can even be extended to white-box investigation of the trained model.

With present day QML solutions, this novel potential attack direction is probably worse, as the service provider scenario will be likely for most quantum computing applications. Indeed, given the principled constructions of backdoored classifiers in [330], there is no obvious reason why the same principles cannot be applied with little to no change in QML as well.

The training process preferably takes place in the cloud (thanks to its high scalability) and due to the fact that using (public) cloud resources follows a more open access approach than using one's own hardware resources (that are not accessible from the web), the possibility of poisoning is greater. With "more open access approach" we mean that the cloud is publicly available to everyone (according to the pay-per-use model). This naturally means that security requirements for public cloud resources are very high, but this does not change their general openness to the public.

> The robustness behavior of QML systems to poisoning attacks remains an open area of research. For instance, if quantum computing is used as a mere tool for accelerating the estimation of the parameters of a classical ML model, it is not obvious that a quantum trained model should be more robust against poisoning than a corresponding classically trained model. If, on the other hand, quantum effects such as superposition are used for novel data encoding prior to training, poisoning attempts might fail immediately or retraining may help.

Barreno et al. [310] suggest the learner should ignore potentially corrupted data as a counter strategy against data poisoning and, respectively, against an attacker's possibility of taking advantage of false confidence. The authors also suggest confusing the attacker who is poisoning the data with adversarial examples by altering the attacker's poisoned data. This way, the attacker's assessment of the learner's condition is being fooled. If the defender is able to automatically detect poisoning attacks and intercept poisoned data, then fooling of attackers may cause the effect that the attacker wastes his attack energy or even loses the desire to (and abandon his) attack. This, of course, requires the system to identify malicious/poisoned inputs. An indicator of such malicious input is the increase of occurrence of data which differs from the pattern of the majority of input data. However, outlier detection is itself a non-trivial problem and might require the integration of additional system components such as autoencoders which can accomplish this task [331]. Considering an intrusion detection scenario, Barreno et al. [310] pick up the idea of tricking the attacker by letting him think that a particular intrusion is not part of the training set, leading to this intrusion being a honeypot. Such approaches seem, in principle, also be possible in the context of hardening QML systems against poisoning attempts as the basic idea is platform agnostic. Clearly, the honeypot principle is applicable to QML by providing honeypot quantum computers. In addition, quantum computing devices may ease the outlier detection in the following way:

Classical ML techniques yield a single class label, or at best, the marginal probabilities of each class. In contrast, each quantum state represents a probability distribution over all 2^n computational basis states. Drawing a specific number of samples (shots) then yields an empirical estimate of this distribution. The outputs, say, different class labels, are hence not independent, resulting in a larger family of probability distributions. We may hence keep track of this distribution by, e.g., computing the average Kullback-Leibler divergence between the estimated output distribution and some reference distribution. The reference may be calibrated at training time, together with the model itself. At runtime, observing a large deviation from the expected output distribution can indicate an ongoing attack.

Wang et al. [332] deal with poisoning attacks including several countermeasures in intelligent networks such as the fifth generation of mobile networks (5G), Internet of Things (IoT) and Software-Defined Networks (SDN). The authors see in ML a measure for defending or protecting such networks. They compare different poisoning attack techniques and discuss poisoning attacks on deep learning networks, linear regressions, SVMs etc. Some countermeasures Wang et al. enlist are in line with the ideas mentioned by Barreno et al. [310]. Additionally, they discuss model verification and gradient shaping. In the latter, extraordinary small or large gradient values are clipped to a user defined range. Moreover, small perturbations are added to robustify the learned model against potential noise injections. Interestingly, both appear in the QML context in a natural way: First, parameters of quantum gate circuits represent rotation angles, having a naturally bounded range of at most 2π radians—the model parameters are hence clipped automatically. Second, various kinds of noise influence the quantum computation. But even in the case of a perfect, noise-free system, results of the computation can only be accessed in the form of samples, generated via quantum measurements. Thus, any output quantity is (at least) subject to some amount of sampling noise.

11.4 Privacy Attacks

Privacy attacks represent attacks on a system's confidentiality. Security in the sense of personal information (Personally Identifiable Information, PII in short), should always be considered in the context of application security. Whether in the environment of classical ML or in the QML domain, the need for protecting personal data remains the same. For example, such an attack on a face recognition model has the effect of violating the privacy of users of various online platforms, since an attacker can use knowledge such as a person's name to compute the approximate image that was used to train the model.

Again, it may be the case that issues such as decoherence, random fluctuations, or measurement noise of current quantum computers could make such attacks more difficult. In fact, defensive concepts such as k-anonymity might benefit from noise inherent to quantum computing processes. However, the question whether QML will lead to fundamentally new kinds of learning based privacy attacks is hard to fathom at this time. As always, if potential learning based attacks on personal information are so costly that they are hardly feasible on digital hardware, quantum computing may provide a tool for making them practical. Yet, with respect to known existing privacy attacks on ML, quantum speedup does not seem necessary.

Rigaki and Garcia [315] examine more than 40 papers that deal with the topic of privacy attacks against classical ML systems and introduce a categorization of privacy attacks, including the two dimensions assets under attack and adversarial knowledge. Moreover, the authors investigate causes of privacy leaks and defenses that are suggested the most frequently. A central type of attack they mention is the so-called *Membership Inference Attack*, where the attacker "tries to determine whether an input sample x was used as part of the training set" [315]. This type of attack was first described by Shokri et al. [333] and discussed as well by Salem et al. [334] who demonstrate the applicability and realizability at ease with result of high effect.

Saki et al. [295] describe the so-called *readout sensing* as a privacy attack where an adversary 'senses' the victim's qubit. More specifically, the attacker first collects reference signatures using a device that runs circuits on both qubits and then compares his qubit based on the statistical distance. The authors consider reading a victim's data as an attack on the victim's privacy.

Privacy preserving ML is a research field that deals with the question of how one learner can train his model without losing his privacy or (in a distributed setting) how more collaborative learners can contribute to train a model without losing their privacy. This field encompasses various approaches that are already used in practice.

In the following, we consider defense strategies rather than attacks in the privacy context.

Privacy-preserving federated learning: this is a form of distributed ML where multiple collaborators train a model through shared protected gradient information [335], [336].

Differential-privacy: approaches in this class aggregate data from collaborators and extract information without loss of their privacy by introducing randomness [337]. Rigaki and Garcia [315] call *Differential Privacy* an established defense method against membership inference attacks which follow the principle of "learning nothing about an individual while learning useful information about a population" which in turn can be technically achieved by building in randomness, so that the result of the algorithm hardly differs if two databases used by an algorithm differ by only one record. Senekane, Mafu and Taele [338] consider differential privacy methods as a countermeasure against privacy attacks on QML. Senekane, Maseli, and Taele [339] explain the concept of Differential Privacy and develop a privacy-preserving QML scheme using a quantum computing framework. In their case, the authors add Laplace noise to the input data, and they successfully validated their method for privacy-preserving QML using the Wisconsin Breast Cancer data set (including features and target labels).

Moreover, in contrast to classical computation, QML methods enjoy a natural degree of differential privacy due to the inherent noise floor of quantum computation. Any data fed into a learning system together with the subsequent computations are subject to quantum noise. Clearly, when not even the learning algorithm operates on the true training data, it is basically not possible that an attacker reads the denoised training data out via some privacy attack. It is an open research question which level of privacy can be achieved by relying solely on inherent quantum noise.

An open QML research question in this context is the quantification of the degree of privacy that could be gained due to the native presence of quantum noise. Such measures are required to understand whether QML deliver a significant difference in the level of privacy and if this difference is practically relevant.

Differentially private synthetic data generation: this term refers to privacy preserving approaches for generating synthetic data sets without having to compromise on the utility of data [340].

Privacy-preserving multi-task learning: methods in this class refer to the idea of secure distributed Multi-task learning (MTL) to protect collaborator's privacy on transfer protocol layer [341], [342].

At the time of writing, it is likely that quantum computers serve as special purpose accelerators, comparable to programmable graphics processing units. Thus, even when large-scale robust quantum computation is available, their usage might be limited to highly demanding computational tasks. Nevertheless, since some data science problems require heavy computation, applications from various fields, included those that rely on private computation, will be carried out on quantum computing devices.

11.5 Side-Channel Attacks

We would like to define the term side-channel (SC) attacks a bit broader than most of the literature and understand it to include the attempt to draw conclusions about the (Q)ML model via hardware-based measurement methods. In existing literature, the SC term has its focus on attacking implementations of cryptographic methods. This happens by finding correlations between the side-channel leakage during the execution of an algorithm and the observed data. A side-channel leakage can be the runtime of the algorithm, the energy consumption of the processor during computations, or electromagnetic radiation. Since hybrid models also use classical devices, classical side-channel attacks are also relevant for QML applications. This is true regardless of limitations of NISQ era devices, since the hybrid approach will likely still be used once more powerful quantum devices become available.

However, if we consider side-channel attacks on quantum computing models exclusively, the situation will be different, because a quantum computer cannot be probed for its internal state since quantum measurements destroy quantum states. Instead, attackers might rely on temperature and cooling control as a leakage for side-channel attack. It should be noted that in general, however, attackers are able to maliciously add qubits and entangle them with the targeted qubits used for quantum computation. This applies to input, output, and even auxiliary qubits used “only” during the computation itself. Another approach could be to try to tamper with error correction. This is because quantum error correction itself is difficult and recent proposals to optimize known error correcting protocols or codes make use of classical ML [343, 344] which may be vulnerable to attacks.

> The issue whether decoherence and noise (that creates opportunities for an attacker to read out information as well) opens a new kind of side-channel, forms an open research question. For example, in parallel to the quantum chip, an attacker could execute corresponding code on classical resources and measure the deviations, which in turn form a kind of side-channel information. In other words, this way an attacker would be able to extract certain characteristics such as noise patterns of the quantum hardware.

Simple Power Analysis (SPA)

In the SPA technique presented by Avoine et al. [345], the attacker tries to determine the secret key from a single trace of side-channel information. This can be done by looking at the difference of the basic public key operations. If the observed signal-to-noise ratio (SNR) is not sufficiently large, the success of this attack technique is unlikely, so that the developed countermeasures usually protect successfully. The success of such an attack is even less likely with QML, since quantum measurements destroy quantum state. Measurable leakage during operations could be temperature and cooling control, which is an open area for further investigation, as stated in the research questions below.

Differential Power Analysis (DPA)

In the DPA technique [345], many traces of different data are examined for the algorithm in question. Then a brute force attack is performed on parts of the algorithm according to the principle of “divide and conquer”. To reduce noise by averaging, many samples are needed. It is unlikely that enough samples can be collected to carry out the attack successfully, as collecting quantum side-channel leakage is very difficult, as mentioned before.

Correlation Power Analysis (CPA)

The correlation performance analysis (CPA) presented in [345] is a multi-bit performance model that reduces the impact of noise on a successful attack. The main difference with DPA is that DPA is based on calculating the difference between two trace sets, while CPA uses the correlation coefficient to calculate the dependency test. With the same justification already given for DPA, it is unlikely that CPA will be successful for QML.

Template Attacks (TA)

Template Side Channel Attacks (T-SCA) are a specific subclass of Profile Side Channel Attacks (P-SCA) where the attacker has access to another, fully controllable copy of the device he is attacking. Such attacks are typically focused on cryptographic algorithms and mainly aim at obtaining cryptographic keys (the secret keys / private keys) from protected systems. Here, we would like to remind again that our conceptual understanding of side channel attacks include along cryptographic systems also (quantum) ML systems as an attacker's target.

Deep Learning for Side Channel Attacks (DL-SCA)

Again, the goal is to exploit leakage to reconstruct a key. Maghrebi [346] utilizes DL-SCA as an alternative to template attacks, which usually intend to find (discover or reconstruct) a cryptographic key based on the output ("leakage") of information the system yields when performing its cryptographic algorithm upon inputting data. Applied to (Q)ML, we speak of discovering or reconstructing a classification or another (quantum) ML algorithm. The leakage information is of physical nature such as power consumption, processing time or electromagnetic emanations. For this, he assesses the efficiency of this technique in realistic and practical situations. He is able to conclude that the robustness of this scheme is sensitive to variations in the parameters like distance in time samples between the points of interest, the dimensionality of the area of interest and the pre-processing of the data (intrinsic characteristics of the manipulated data set). In contrast, DL-SCA is not sensitive in respect of the leakage model function and the addition of artificial noise, in fact variations in these parameters can maximize the robustness of the scheme. Magrebis investigations allow him to conclude that DL-SCA is very efficient even if the implementation of the cryptographic or (quantum) ML algorithm combines different side-channel countermeasures.

Having now carried out a few thoughts on how known side-channel attacks can be applied to QML, let us now turn to a slightly different scenario. Classical ML or QML can be used to optimize side-channel attacks on classical IT and on post-quantum cryptography.

Marzougi et al. use ML similarly, and attack a post-quantum public-key cryptographic algorithm [347]. In the current implementation of the Lattice-based scheme "bliss", a subroutine proved a target for power side-channel analysis. The authors make use of a profiling phase, where the computing device is analyzed, and an ML model is trained. The subsequent (or even preceding) application of the cryptographic routines are then analyzed by the classifier, and results are used for key recovery. As classical ML improves such side channel attacks, an open question is whether QML methods can be made into a promising counter measure—or an additional threat—for post-quantum cryptography.

Das et al. [348] present a new Cross-device Deep Learning Side-Channel Attack (X-

DeepSCA), which significantly increases the threat surface for devices implementing a crypto algorithm, which the attacker seeks to profile. A profile is being created by using a Deep Neural Network (DNN) that learns from augmented traces (power side-channel leakage) from not just one, but multiple devices. Previous work evaluated and trained the attack on the same device, potentially leading to overfitting and discrepancy resulting from variational leakage between devices. This new approach allows for more robustness in the model. The presented attack is characterized by an accuracy of 99,9%. Kashyap et al. [349] go a step further—they confirm and demonstrate classical ML as an enabler for side-channel attacks on post-quantum key exchange protocols. The authors introduce *2Deep*, an approach to enhance Side-Channel Attacks on Lattice-Based post-quantum key-exchange (PQKE) protocols, and they emphasize the importance of researching appropriate countermeasures.

Gohr, Jacob and Schindler [350] attack AES implementation using a new Deep Learning architecture for side-channel analysis and demonstrate how deep neural networks outperform existing solutions substantially by reverse-computing the subkey bytes' Hamming weights of AES round keys at a high success rate.

Practical implementations or even optimizations of side-channel attacks on Q(ML) do not yet exist. Although it seems hardly implementable in practice at the present time, side-channel attacks on QML opens up a field of research in which developers transfer attacks aimed at cryptographic systems to attacks against QML systems.

> Side-channel attacks such as SPA, DPA and CPA do not form a serious threat to QML, since quantum measurements destroy quantum states. The only measurable leakage during operations could be temperature and cooling control, however, it is questionable and an open question to investigate whether this kind of information is useful at all.

When we assume that an attacker has the ability to modify a quantum circuit only by injecting additional gates that work on additional auxiliary qubits, computation in the original circuit can be disturbed. Even when the newly added qubits and gates do not at all interact with the original qubits, crosstalk can affect parts of the latter [351, 352]. In fact, having knowledge about the hardware may allow for highly effective crosstalk attacks. As stated in Appendix C of [352], each qubit of a superconducting circuit is physically connected to the chip via a separate control line. Ideally only the addressed qubit will react when its control line is used. However, due to unwanted hardware imperfections, a small fraction of the signal from any line will reach several qubits. This effect is called *linear flux crosstalk*. Technically, when one measures the qubit, one may observe a deviation from the phase even in the absence of a signal on the source line. That change is hence induced by the control line of the qubit itself or a neighboring coupler. It turns out that lines physically closest to one another have crosstalk around 0.1-0.3%. An attacker that has access to the circuit might hence induce crosstalk related errors by frequently accessing neighboring qubits and qubits with nearby control lines. An appealing property of this novel attack design is the fact that the gates added by the attacker are not connected to the original circuit. The user will thus measure elements from the intended Hilbert space without observing the additional malicious circuit parts.

The physical and mathematical behavior of these crosstalks and the concrete impact on victims that are attacked by adversaries having access to the circuit opens a new field of research, which can be investigated more deeply.

Can QML improve DL-SCA?

Earlier we have described side-channel attacks that utilize classical ML, or more specifically modern Deep Learning (DL-SCA), as a tool. It is an interesting question whether the usage of QML would give an attacker an advantage over ML. We interpret based on [353] that a QML model would need less training data or simpler models to achieve the same learning effect. This leads to an improvement in learning efficiency, which in fact gives an attacker an advantage and is especially a problem when security depends on an attacker not being able to collect "many observations." This is another field where we see an unresolved need for further research.

11.6 Other Attacks

Saki et al. [295] list various other attacks that might affect QML directly. First, *crosstalk-induced fault injection* is mentioned, which "exploits crosstalk errors for fault injection attacks in a multi-tenant computing environment where two or more quantum programs can run simultaneously on different physical qubits". The crosstalk-induced fault injection differs from the approaches behind side-channel attacks in the sense that an attacker finds and exploits weaknesses in a multi-tenant architecture. Thus, the special situation here is that two "circuits" are used *in parallel* on one quantum device, which in the words of Saki et al. "is economically enticing as it maximizes hardware resource usage and profit".

The next attack refers to the fact that quantum circuits are sent to quantum hardware via a cloud. Since the user has no insight into the hardware used, an attacker could —depending on his ability to influence— assign the user worse quantum hardware than expected, so that the user gets results of a different quality than expected.

There are many derivatives or modified methods of the attack approaches presented so far. Not all methods can be unambiguously categorized into our categorization scheme that uses the categorization dimensions taken from existing literature (such as for example white box, grey box, black box). In the following, we present a few adapted (specialized) types of attacks including model extraction via the (Q)ML Application Programming Interface (API), which originate from model stealing attacks described in Section 11.1.

Kong et al. [354] give a detailed overview of adversarial attacks and classify them in a way that is very similar to ours. Among other things, they consider the previously unmentioned malware-based attack, which attacks malware detection methods. In addition, physical attacks are mentioned where an attacker does not understand the structure of the model and even has weak control over the input, thus representing another category for the authors in distinguishing attacker knowledge (besides black, grey and white box). From Kong et al. point of view Malware is malicious code that poison training and test data by adding disturbance to samples, or collecting information about a target model.

Model stealing has already been covered in Section 11.1, in the following we will focus on a specific case - that of the cloud environment and API usage. Tramèr et al. [355] demonstrate model stealing (model extraction) attacks against ML cloud services that extracts the ML models with high accuracy for widely used model classes including logistic regression, neural networks, decision trees and SVMs. The starting point for these attacks is black-box access to the target model ML without prior knowledge of the model's parameters or training data. The authors highlight that the natural countermeasure of setting API restrictions to prevent model extraction, such as omitting confidence values from model outputs or restricting inputs, does not eliminate all potentially harmful attacks, as they are able to present a slower but still dangerous attack that can extract the model based only on class labels.

In addition, QML APIs should ideally not expose information about the underlying low-level hardware that is used to train and apply the model. As explained above, knowledge about low-level properties of the quantum system may allow for very specific attacks without any classical counterpart. Non-exposure of hardware details, however, can also be a limitation whenever users might want to work with these specifications for their own quantum circuits. This can be of interest because, just as on digital computers, there often exist several possible circuit layouts to realize a certain quantum routine. Some of these will be more efficient than others, and corresponding low-level information may be required to tune a circuit towards aim such as efficiency or quantum noise robustness. On the other hand, in a scenario where users are only allowed to run pre-defined QML circuits, it is reasonable to assume that significant hardware details can remain hidden because the transpiler offered by the quantum computing service provider takes care of circuit optimization. This, however, could lead to questions as to the security or vulnerability of transpilers.

> Can it be guaranteed that a transpiler, i.e. the component of a quantum computing ecosystem which maps a given circuit specification to the topology of a quantum device, performs reliably and is shielded against attempts of tampering?

Frederikson et al. [356] focus on API usages as well. They present new model inversion attacks that can be used to derive sensitive features from decision trees in ML services or to extract images of training subjects from face recognition models. They take into account commercial ML as a service APIs, where adversarial clients attempt to perform model inversion attacks using confidence levels that are disclosed along with predictions. They experimentally demonstrate on real data that these attacks pose a significant risk to feature confidentiality, and discuss initial countermeasures that could mitigate the presented attacks and perhaps prevent future ones.

Programming interfaces for QML will be available analogously to classical ML interfaces. To some extent, such QML frameworks already exist and exhibit a certain maturity. A vulnerability is therefore given similar to classical ML—the only difference being that free public solutions and even commercial cloud-based solutions do not currently offer the performance that would allow such API based model stealing attacks to be efficiently implemented. At the appropriate time, when high-performance quantum resources are made available via APIs, these attack approaches will gain importance.

> Any quantum computation in current NISQ devices is subject to various sources of noise. Moreover, algorithms and QML methods must be designed with the awareness of perturbations. They require implicit or explicit mitigation techniques to ensure reliability in the presence of noise. It is, however, an open research question whether this also implies that specific attacks on QML systems require an increased level of noise robustness, too.

12 Additional Defense Strategies

At the time of writing, and even into some time in the near future, the described attack strategies on quantum systems will not be of any factual practical value. However, this will gradually change as quantum devices and quantum computing become more powerful. Therefore, both in research and in industry, the capabilities of quantum computing evolving must be followed closely. Finally, by the time quantum computers will be able to deliver considerable performance, the implications for security in classical computing are substantial and to be taken into serious consideration.

Accordingly, we now discuss how QML techniques could perhaps lead to novel defense strategies and more secure solutions. Along with explanations of attack strategies, various thoughts on respective defensive measures or responses have already been included in section 11. In this section, additional quantum-based protection efforts will be discussed. These are based on research that specifically takes the point of view of the attacker.

A very general, introductory observation regarding the latter is that inherently secure and reliable solutions are, naturally, less vulnerable and thus require fewer defense efforts. Here, it is interesting to note that due to the increased proliferation of artificial cognitive systems in real-world applications, questions as to robustness and explainability of their results are becoming increasingly pressing [293]. This is not only because incorrect or non-transparent decision making can have dire consequences. This raises numerous still unanswered issues about accountability and related legal topics [357]. The research communities working on classical ML have therefore begun to develop methods and criteria for more trustworthy, accountable, and transparent intelligent systems. Their general consideration and first attempts at standardization should, in general, also apply to the design of quantum computing systems for ML. However, as quantum computing and QML are still very much nascent fields, larger scale research efforts, and corresponding dedicated literature on robustness, explainability, or certifiability of QML solutions do not yet seem to exist. As we will discuss below, existing considerations mainly apply to robustness as a technical issue (for example to quantum noise and decoherence) rather than to a societal or legal one.

> Do legal requirements necessitate novel concepts or criteria for robustness, explainability, or certifiability of QML systems? Are existing concepts and criteria developed with regards to classical ML systems perhaps sufficient and transferable?

Lu, Duan and Deng [304] introduce strategies that counter adversarial examples. The authors point out that there is no universal strategy, but each countermeasure against adversarial QML must be worked out on a case by case basis, specific to each attack type. One approach to enhance the QML robustness is *quantum adversarial training*, where defenders inject adversarial examples into the training set. Moreover, the authors refer to defense strategies aimed at protecting classical ML systems including strategies such as "gradient hiding", "defensive distillation", and "defense-GAN". They propose to transfer these into the world of QML by implementing, for example, a "defense-QGAN".

Examining the question of whether a QML learner can perform learning tasks securely, Song et al. [358] introduce a "probably approximately correct" (PAC) learning model using a classical-quantum hybrid sampling protocol that enables QML being secure against maliciously manipulated learning samples. The authors define a security constraint that rules out adversaries and allows only legitimized learners to sample a finite set of input data, ensuring effective learning outcome. They demonstrate a lower bound for the learning samples ensures PAC learning and an upper bound for blocking intruders. Finally, to enhance quantum security and privacy simultaneously, Suresh et al. [359] propose a quantum circuit obfuscation method that hides the true functionality of a circuit by inserting dummy gates. This makes it more difficult for attackers to steal quantum learning models and logic.

García de la Barrera et al. [360] consider the nascent research area of quantum software testing (QST) and provide a systematic literature review on this field, including probabilistic testing, Hoare logic (a fundamental concept in theoretical computer science) and reversible circuit testing. They observe that a growing number of researchers is becoming aware of the need for concise QST, and that established frameworks do not yet exist. This seems to offer opportunities for further research. Especially since computational methods for verification are often burdensome or intractable on classical computers, novel quantum-based approaches could offer new tools in this regard.

> Quantum computing allows us to implicitly work with exponentially large state spaces. If used appropriately, this capability might pave the way for automated program verification, especially based on logic representations whose solutions are classically intractable. This raises questions as to which extent quantum computing and QML techniques could contribute testing procedures for classical- as well as for quantum-based software.

12.1 Defending Attacks on Privacy

Due to the high cost and complexity of quantum computing, most users will access QC through a cloud server via Quantum as a Service (QaaS). This public accessibility of quantum resources leads to an increased need for protection and security. The threats associated with the use of quantum computing have been discussed in section 11, where various methods of defense have also already been visited. As QaaS also implies that users train quantum models on a remote quantum device using their (private) data. As a consequence, privacy comes into focus and the privacy attacks are of special importance in the defense context.

A recent line of work in QML addresses privacy attacks from a genuine quantum computing point of view. For instance, Watkins et al. [361] point out that privacy-preserving algorithms are not implemented in current QML models. They demonstrate a QML model that protects sensitive user data by building the model with a differentially private optimization algorithm. That is, an algorithm that minimizes the impact of a single data point in the training data set. Although their proof-of-concept is only performed via simulations on a classical computer, they consider their results an indication that their model can be efficiently implemented on current NISQ devices.

Senekane et al. [338] present a privacy-preserving ML method based on logistic quantum regression that uses discrete laplacian noise as a noise model for anonymizing perturbations at the stage of input. They evaluate their method on a set of sensitive medical data and observe high accuracy, high precision, and high recall and thus establish that privacy-aware approaches are possible on quantum devices.

Another quantum-based countermeasure against privacy attacks is *distributed secure quantum machine learning* (DSQML) as proposed by Sheng and Zhou [362]. It suggests augmenting a classical client with certain quantum technology to delegate remote QML processes to a local quantum server to preserve privacy data. Their work describes a DSQML protocol that is capable of processing big data and classifying high-dimensional data vectors. The protocol is described as secure in the sense that it does not leak relevant information and, any intruder trying to silently intercept the learning process will be recognized.

Similarly, Bang, Lee and Jeong [363] introduce a protocol for secure QML being executed at a remote location. An arbitrary initialized device at a learner's location is trained by a provider at a remote location. The authors design their protocol such that any external learner trying to join or perturb the learning process will be detected. The authors show numerically that their protocol operates reliably for devices with single-qubit operations.

Using a quantum convolutional neural network (QCNN), Yang et al. [364] introduce yet another decentralized learning process, and they consider an application in speech recognition. Their federated learning process works with quantum convolutional layers distributed across several NISQ devices and ensures privacy-preservation for speech data. This may, for instance, be useful in securing biometric-based access control. Also working with QCNNs, Chen and Yoo [365] examine the concept of federated QML on a more general level. They develop a federated training approach for hybrid quantum-classical ML models which can be generalized to pure quantum ML models. As these authors, too, are concerned with speech recognition problems, their techniques, too, appear to be useful in sensitive applications. In either case, however, an obvious quantum advantage over classical ML solutions is not immediately apparent.

Coining the term *blind quantum machine learning (BQML)*, Zhou and Qiu [366] introduce another distributed quantum protocol that enables a remote learner (with limited quantum capabilities) to delegate his QML task to a quantum server (with extensive quantum capabilities) such that the remote learner's data is kept private. However, their approach is still of rather theoretical nature and, in addition, assumes the existence of QRAMs which, at this point in time, are not technically feasible.

12.2 Robustness and Noise of QNNs

Specifically dealing with the security of intelligent systems based on neural networks, Wang et al. [367] present a framework for QNN training that takes quantum-specific noise into account, both in the training and in the application phase. Their framework essentially combines three ideas which, as the authors demonstrate, significantly improve the accuracy of the model. First of all, they regard the experimental observation that there is a linear dependency between a QNN measurement result and the noise-free result that depends on a scaling- and a shift factor. This motivates the idea of post-measurement normalization. The second idea is noise injection, where quantum error gates are added according to realistic noise models of quantum hardware during training. Finally, they propose post-measurement quantization, in which they make quantitative measurements to remove noise. All in all, their results indicate that QNNs can be made more robust and less vulnerable than commonly thought.

Katzir and Elovici [368] specify a *model robustness score* for measuring the resilience of ML based classifiers that are employed for cyber security purposes. To elaborate the resilience, especially against adversarial attacks, the authors formalize their quantification model and provide a corresponding implementation which measures the necessary amount of effort that is needed to manipulate ML. Even though the authors do not refer to QML but to classical ML, their ideas for specifying a so-called *model robustness* are transferable to QML.

Another interesting aspect of QNNs, especially with quantum convolutional layers, is a recent empirical finding. Reese et al. [369] consider the use of such quantum convolutional layers in a vision-based quality control scenario and observe that quantum encoded input data can—at least in their use case—be more expressive than classically computed representations. That is, they observe that a hybrid quantum-classical convolutional neural network can learn more robustly from less data than a classical CNN. From the point of view of preventing adversarial attacks, it may thus be possible that such systems could better cope with adversarial manipulations of the input data. Indeed, adversarial attack detection has traditionally revolved around CNNs for computer vision, so that these findings could lead to new baselines. However, this certainly requires further study.

Can quantum convolutional layers (for now still integrated into classical CNN stacks) generally learn more expressive representations than classical convolutional layers? Are corresponding reports in the literature confined to very specific kinds of input data?

> Moreover, if there exists an entanglement-based quantum advantage for representation learning, will it lead to systems that are inherently robust against adversarial inputs?

12.3 Defenses of QML Based on ML

Existing literature has not yet addressed possibilities of using classical ML as a protection tool to identify or counter attacks against QML. However, there exist approaches to attack quantum key distribution (QKD), see for example [370], or [371], and there is research available describing how classical ML can be used to protect against attacks on (QKD). In this section, we give ideas on how the use case of protecting QKD using classical ML can be transferred to the use case of protecting QML.

Quantum key distribution (QKD) refers to methods of quantum cryptography that use properties of quantum mechanics to provide two parties with a common random number, which they use as the secret key. Therefore, in the use case of QKD, the artifacts worth protecting are the key and the algorithm behind it for generating that key. To apply this to QML, we could consider QML devices as the artifact worth protecting.

For instance, Mao et al. [372] are concerned with continuous-variable quantum key distribution (CVQKD) systems which are vulnerable to a number of different attacks (Trojan-horse attacks, wavelength attacks, calibration attacks, local oscillator, intensity attacks, saturation attacks, homodyne-detector-blinding attacks) for which no generic/universal defense strategy exists. These attacks are mostly based on the basic idea of a “man in the middle”, who situates himself between two communication partners and intercepts/reads the communication stream and takes advantage of the shortcomings of quantum devices that consists in the attacker’s ability to hide themselves by introducing little noise. The problem lies in the hard challenge to estimate the excess noise and thus to detect noise discrepancies caused by such little noise that attackers have injected.

The authors suggest how to counter such attacks using classical ML. But what can we deduce for (quantum) machine learning? In the case of QML, we have also an eavesdropper principle and also the approach that adversaries introduce noise / perturbation data into the training process of the model. An idea would be, using classical ML such as for example classical anomaly detection, to unveil hidden perturbers that manipulate a QML system.

As Mohammed et al. [373] describes a similar case, with close regard to IoT. The authors state, the limited power and computing capabilities of IoT devices are a major challenge in designing and implementing their security. This becomes even more acute in the age of quantum computing, as attackers could rely on quantum computing capabilities, making IoT devices even more vulnerable. With reference to QKD in 5G networks, the authors develop an algorithm to detect an attacker interposed between sender and receiver, with the side effect of the QKD process being disrupted during the intruder discovery. Furthermore, the authors utilize neural networks and deep learning to recognize an intruder during QKD without having to interrupt the key distribution process.

Again, using classical ML, the idea of revealing an unwanted party could be applied to QML. On one hand, these are the infiltration, manipulation of data (in terms of the previous paragraph) for a simple local learning setting; and on the other hand, these are distributed (QML) learning processes where data or gradients are exchanged via quantum protocols with an attacker eavesdropping or interfering. The latter seems particularly suitable to establish for a possible transfer of ML-based defenses from QKD to QML. This is possibly a starting point for a dedicated emerging area of research.

Liu et al. [374] design an integrated support vector regression (SVR) model for improving both a QKD system's performance and its operational security. In the first step, the authors train an SVR model to accurately predict the temporal trend of the physical signals parameters. Then, they use this trend as feedback to control the QKD system for optimal performance and security. Based on their experimental results, they point out three main advantages of their approach: First, the QKD system achieves optimal performance and security. Second, they do not need a real-time monitoring module for predicting the trend of physical parameters of signals. Third, their model is adaptable to any measurable physical parameter of signals in a practical QKD system.

The feasibility of predicting temporal physical signals parameter trends and predicting excess noise in order to protect a QML system opens up a currently uncharted, unoccupied field of research of its own.

From the question how approaches to ML-based protection of QKD (where the secret key and algorithm for its generation are artifacts worthy of protection) can be applied to the protection of QML opens a research field that has not yet been engaged.

13 Conclusion and Outlook of Part II

Given that quantum computing has become technically feasible and promises accelerated computations for demanding machine learning tasks, it is no surprise to see that research and development in the area of quantum machine learning are growing rapidly. However, general characteristics of quantum computing as well as technical limitations of present day NISQ devices raise questions as to the vulnerability of QML methods. We therefore fathomed possibilities of malicious attacks on QML systems and possible defense mechanisms or countermeasures. In either case, we focused on practically feasible scenarios and solutions rather than on still purely theoretical QML concepts.

Guiding questions were: Does the attack surface of QML systems and hybrid quantum-classical learning systems increase? In other words, does the nature of quantum information processing turn QML systems into potentially easier targets than classical ML systems? Do attacks become more effective when they rely on available quantum technology? Or, in the opposite, do become defense mechanisms more effective when based on quantum technology? Which side, attacker or defender, is more likely to benefit from potential quantum advantages? Under which conditions may either side see a greater increase in effectiveness?

In an endeavor to answer these questions, we looked at where and how known attacks on classical ML systems transfer to the QML setting and at what kinds of attacks are conceivable at which step in the learning pipeline. Following best practices of IT security research, we considered dimensions of attack types on QML systems and surveyed the current literature on machine learning specific attacks in the context of QML systems. Examples of such learning specific attack types include model stealing attacks, adversarial attacks, data poisoning attacks, privacy attacks, side channel attacks, as well as attacks on quantum computing as service solutions.

Given the current state of the art of quantum computing systems, our analysis necessarily involved a foresight perspective. Put differently, in most cases there are no definitive answers yet. Common themes found in the current literature are general statements such as that known types of attacks on ML systems should be transferable to QML systems, that quantum noise may be a boon or a bane for QML security, or that quantum decoherence due to measurement may provide a native security mechanism for QML systems. In short, our systematic survey revealed more open questions than definitively answered ones. Put differently, as of this writing, the security of QML systems is still an under-researched area and there are many directions for systematic future research. To be specific, the following lists promising research questions pertaining to the security of QML on current NISQ devices which dedicated research may answer in a near- to mid-term time frame:

Model stealing and privacy in AQC: At present, commercial access to adiabatic quantum computers is mainly offered in form of web services where customers upload the specification of a QUBO and then receive its solution. However, preliminary research indicates that QUBOs are reversible in the sense that malicious attackers may analyze API traffic to infer the type of QUBO being solved and thus to infer details as to the business of the customer. The extent to which this is really possible seems testable, i.e. future research could explore in how far the inputs to an adiabatic quantum computer allow for deducing what kind of practical problem they encode; given that AQC may soon play an increasing role in sensitive sectors such as the financial services industry, this topic may have high priority.

Vulnerability of federated QML: federated or distributed QML systems may potentially overcome limitations such as limited qubit count or limited qubit connectivity that are characteristic for present day NISQ devices. However, pooling the resources of multiple NISQ devices requires communication and data transfer between the participating systems and thus opens the possibility of eavesdropping and poisoning attacks. The extent to which this may be possible seems not to have been investigated yet and should be testable in a straightforward manner.

Adversarial attacks on quantum (neural) networks: compared to classical error backpropagation methods for neural network training, error propagation in quantum gate networks is constrained or limited by the fact that quantum mechanical operators (quantum gates) are reversible unitary operators. Whether or not this may limit the possibility of adversarial attacks on QML systems has not yet been investigated thoroughly.

Adversarial attacks on variational QML: variational quantum computing combines quantum- and classical processing steps. Whether or not known characteristics or measures of the expressiveness and entanglement capabilities of variational quantum circuits allow for a systematic assessment of their susceptibility to adversarial attacks might be systematically evaluated in a dedicated study.

Data poisoning attacks on QML: Just as classical ML systems, QML systems are vulnerable to data poisoning attacks. However, due to the inherently probabilistic nature of data representation on quantum devices as well as due to other kinds of noise present in such devices, it seems that poisonous signals would have to have considerably high amplitudes in order to negatively affect computations. However, this reasonable assumption has not yet been investigated rigorously and further research seems warranted.

Privacy due to quantum noise: similarly, quantum noise and decoherence may provide implicit defense mechanisms against privacy attacks. An open question is, whether one can gain a certain level of privacy due to the native presence of quantum noise.

Utility of side-channel attacks: again similarly, side-channel attacks such as SPA, DPA and CPA do not seem to pose serious threats to QML systems because any quantum measurements lead to noticeable quantum state collapse. The only apparent measurable leakage during the operation of a quantum computer are temperature and cooling control. Here, one would have to investigate if such measurements provide useful attack information at all.

Software testing and QML: as of now, quantum software testing is still a nascent area of research. The extent to which QML could contribute to software testing methods (including penetration tests) for classical- as well as for quantum-software therefore still needs to be researched and validated.

Looking at this list, it becomes clear that "the jury is still largely out". The disruptive potential of quantum computing technologies in general and quantum machine learning in particular does of course raise reasonable concerns with regard to information security. Indeed, the importance of questions such as the above can be recognized by the fact that trade sanctions on quantum computing equipment have been imposed by the European Commission for the first time in 2022 [375], while the United States have sanctioned specific companies in multiple countries in order to limit availability of quantum computing technology for nuclear weapon programs [376].

However, due to the technical limitations of present day NISQ devices and their therefore limited problem solving capabilities, it cannot yet be clearly recognized whether any kind of regulatory measures would have to be instated in order to guarantee the secure use of information technology in the emerging quantum area. Especially with regard to the security of QML, the picture is not yet clear, as research and development on secure QML are still in their infancy. It therefore seems warranted to continue to carefully monitor technological progress and to set up systematic technology scouting processes which keep track of ongoing and future developments. Further suggestions pertain to systematic foresight processes which look at how different scenarios for the future of quantum computing and quantum machine learning could look like.

For instance, as of this writing, it seems plausible that it will still take time until quantum computers become commodity technology. Until then, quantum cloud computing or quantum as a service offerings are the most likely scenarios for how non-institutional or non-governmental users may access quantum computing facilities. This, in turn, puts responsibilities on quantum computing suppliers and providers to offer secure access and to safeguard against misuse. Such responsibilities, however, do not seem to exceed the current ones of providers of conventional cloud computing solutions.

Nevertheless, dedicated funding programs could help accelerate research on the security of QML in particular. Such programs could call for theoretical or empirical results on the possibility or impossibility of various attacks on QML systems. Ideally, they would especially ask for prototypical demonstrators which illustrate robustness or vulnerability of present and future QML solutions. Given the present quantum computing ecosystem, this, in turn, would mean that developers of quantum computing algorithms would have to work in close collaboration with providers of quantum computing hardware. Corresponding programs should therefore allow for and facilitate collaborations between industry and academic institutions.

Given the present state of the art as revealed in this study, priority—if any—might be given to in-depth examinations of model stealing-, adversarial- and data poisoning attacks on QML. This does not mean that, say, privacy attacks would not constitute an important topic for further investigations. Rather, the suggested prioritization mainly takes into account what kind of investigations appear to be practically feasible and what kind of hypotheses appear to be thoroughly testable in dedicated short term research projects.

Part III

Quantum Machine Learning in Cyber Security

14 Quantum Machine Learning as a Tool in Cybersecurity

Previously, we reviewed theory and practice and the interplay of *quantum computing* and *machine learning* (ML). That is, we reviewed the prospects of *quantum machine learning* (QML) where our point of view was directed by technical capabilities in the *noisy intermediate scale quantum* (NISQ) era. We saw that the potential advantages of quantum computing currently cause excitement and expectations regarding the prospects of quantum machine learning, i.e. regarding the use of quantum technologies and algorithms at various stages of the machine learning pipeline. Indeed, the latest reports on progress with regard to possible physical realizations of quantum computers [377, 378] and realizable quantum advantages [379] suggest that developments in this area will continue to accelerate. However, general characteristics of quantum computing (reversibility, probabilistic measurements) as well as technical limitations of present day NISQ devices (short decoherence times, restricted circuit depths, inherent (measurement) noise) also raise concerns as to potential vulnerabilities of QML methods as current devices inherently suffer from reliability issues. We therefore also surveyed QML from the point of view of IT security and discussed various possible malicious attacks on QML systems as well as possible countermeasures and defense mechanisms.

In this chapter, we assume yet another point of view on QML and IT security and investigate the potential role of QML as a tool either for attacking IT systems or for fortifying cyber security.

Given these general topics, it is rather obvious that they should ideally be addressed from the perspective of more advanced quantum computing technology than is available today. This is because currently available quantum computers already illustrate the disruptive potential of quantum computing solutions in a wide variety of applications, but their technical capabilities are still too limited to be of immediate practical use at larger scales. However, while present day quantum computers may not yet pose neither a boon nor a bane for IT security, the technology develops quickly and future quantum computers may play a role in this arena. In what follows, we will therefore, at places, attempt to speculate about scenarios which appear plausible in a world where quantum computers can manipulate many logical qubits, can apply long sequences of quantum gates with large fan-ins without decoherence, and do not suffer from measurement noise. Whenever we attempt such speculations, we will assume that the quantum encoding of classical input data and decoding of quantum output data for classical post-processing is possible in a reliable and efficient manner. In short, we will, at places, speculate about scenarios where (almost) universal quantum computers for reliable large data processing are available at reasonable prices and where there is widespread expertise on how to program or how to use them.

To begin with, we will discuss the role of classical machine learning in IT security and where and how it can be employed for attacks or defense mechanisms. This discussion will form the basis for the subsequent investigation of QML for cyber security. In order to guide this investigation, we will consider several specific scenarios which have been selected based on present IT security considerations and on a review of the related literature to date. To be specific, the scenarios we consider are spam message detection, malware detection, and security based on analyzing user behaviors or network data. We will analyze these scenarios from the point of view of an attacker as well as from the point of view of a defender and look at how QML may influence their goals or the set of tools they have at their disposal.

Before discussing the relevance of Machine Learning and QML as a tool, we discuss cyber security, present the roles of attacker and defender, and present a multi-phased model for complex cyber attacks, the *cyber kill chain* in table 2.

14.1 Information and IT-Security

Information Security targets the protection of information regardless of its representation, e.g. on paper or on hard disk. The protection goals of information security are:

- **Confidentiality:** Data and resources must be protected against unauthorized view or access.
- **Integrity:** Data must be protected against unauthorized changes to ensure it is correct.
- **Availability:** Data and resources must be accessible to authorized users.

Cyber Security is a state in which risks induced by the use of IT because of vulnerabilities and threats are reduced by measures to an acceptable degree. In this way, *cyber security* is the state, where confidentiality, integrity and availability of information and IT are protected via appropriate measures.

The three goals confidentiality, integrity, and availability (sometimes abbreviated as the *CIA triad*) are key in all security considerations. A security concept covers concrete threat scenarios that are mapped against the protection goals. Such a threat can be, for example, a sophisticated ransomware that encrypts data, leading to a loss of availability. It can also be a malicious actor making company data accessible to a competitor, violating confidentiality. In turn, security controls are defined and implemented to support the protection goals, for example a network firewall denying unauthorized access.

Dependent on the aim of a security concept, additional goals might be incorporated, sometimes security goals might even be mutually exclusive. An example can be the balance between anonymity and non-repudiation.

14.2 Attacker and Defender

Scenarios like technical failures and unintended, perhaps rare disaster situations (force majeure), are threats to the confidentiality, integrity, and availability of computer systems. However, the majority of threat scenarios result from threat actors with a variety of motivations, such as espionage, organized crime, hacktivism, or even cyber warfare. These are cyber attacks in the truest sense.

These attacks, independent of their motivation, follow common patterns, and are separated into multiple phases. A common model for attack phases is the so called *cyber kill chain*[5], depicted and discussed in table 2. This model has further evolved into a de-facto standard applied in industry and academia, the *Mitre ATT&CK framework*[6] (Adversarial Tactics, Techniques, and Common Knowledge) with a total of 14 phases / objectives of the attacker.

These phases are relevant from the attacker's point of view, as they represent the necessary steps to achieve the objective of the attack. Similarly, they are relevant for the defender, as each phase poses an opportunity to prevent or detect an attack and react to it. The model is used for multiple purposes, for example, incident reports are often structured following this model, and cyber security technology can be mapped against the model to determine which phases and associated attack techniques are covered by prevention or detection.

The attacker and the defender compete in a fight of successfully attacking against preventing or detecting an attack. This competition has lead to a whole industry on both sides, each creating sophisticated tools to support the respective side's objectives. Due to the nature of attacked systems, some of these tools are used by both sides.

Among others, attackers use defender's tools such as tools applied in Red Teaming exercises. These tools support in the various phases, e.g. weaponisation, installation, and command and control.

The defenders, on the other hand, have a much different scope, and need to ensure defense means for a broad range of potential entry points. They have to apply a myriad of technologies, to name a few:

- email gateways that filter out harmful emails and spam,
- endpoint protection platforms (EPP) to detect and prevent malware,
- or security information and event management (SIEM) systems to collect security-relevant logs from enterprise assets and the underlying networks to identify malicious activity.

[5] `https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html`

[6] `https://attack.mitre.org`

The level of sophistication in the competition between attacker and defender, as well as the amount of data available for collection and analysis, naturally motivate the application of machine learning in cyber security.

Phase	**Attacker**	**Defender**
Reconnaissance	Conduct research on the target—passively e.g. in the Web or Darknet or actively e.g. via network scans	Active Scans can sometimes be detected on a network level, passive research is even more difficult to detect, e.g. via Darknet monitoring
Weaponisation	Develop or adapt a tool for the attacks, e.g. a malicious document	This phase is outside the control of the defender
Delivery	Deliver the "weapon", e.g. email the malicious document to victims determined during reconnaissance	Detect and defuse the malicious mail
Exploitation	Gain privileges for installing malicious software (malware) on systems of the victim, e.g. via exploiting a vulnerability	Fix vulnerabilities, detect and deny malicious activity
Installation	Install the malware	Prevent installation, detect and delete malware
Command and Control	Remote control of the victim's systems	Detect and prevent control
Action on Objectives	Reach the goal of the attack, e.g. collection of confidential data and exfiltration	Detect and prevent the attacker activity, e.g. through a firewall

Table 2: Cyber Kill Chain - a model for sophisticated cyber attacks

14.3 Machine Learning in Cyber Security

In their review, Dasgupta et al. [380] provide an overview, and present how ML has been used for the purpose of defense in cyber security. They approach the subject from a general standpoint and thus provide extensive information on currently used ML methods. They also portray some of the available cyber security data sets that can be used to train and benchmark models. The authors narrow down their scope to significant work of the six years from 2013 to 2018, noting that current research has not yet deeply investigated the threats that affect ML models that are used for cyber security. Going further, this would be a basis and requirement for any investigation of defense methodologies and approaches.

The authors mention many ways to use ML methods for classification or detection in cyber security. One of the simplest mentioned ML method is a decision tree (DT) which can be used to distinguish between legitimate and illegitimate activities when examining network logs. DTs provide high accuracy, but suffer from high space complexity. Further, a DT can be combined with a support vector machine (SVM) and a Naive Bayes classifier to form a “collaborative classifier” with high overall accuracy. Another useful ML classifier is an advanced neural network for identifying shellcode patterns in network traffic data. Finally, SVMs can be applied for network intrusion detection.

Naik et al. [381] review literature on multiple artificial intelligence techniques differentiating between “distributed” and “compact” methods (ML as part of the latter) on the topic of cyber security outlining the defender’s perspective. The authors deduce from their analysis that the discussed methods do have a major impact for cyber security. The covered cyber threats mostly relate to (network) intrusion detection and spam/phishing classification. An interesting piece of this work is the discussion of possible challenges in applying the techniques. The authors point out that the attackers are able to use the same methods, and, for example, train a classifier to identify vulnerabilities suitable for exploitation or adaptable malware.

The identification of vulnerabilities by means of an analysis of the source code (in contrast to the detection via execution of the code) is the focus of a recent study of the Federal Office for Information Security, where the authors investigate ML in the context of static application security testing (ML-SAST) [382]. Static application security testing augments secure software development and represents a technology to reduce vulnerabilities becoming part of an exploitable software stack. In this way, the attack surface is reduced and the exploitation phase becomes more difficult for the attacker. There are different ways to approach SAST, prevalent, according to the authors, is a transformation of the code into an intermediate model denoted as abstract syntax tree.

As stated by the authors, an uptake in research publications in this area is notable in the last years. Their literature review led to the identification and discussion of five core technologies suitable for ML-SAST: graph neural networks, conventional neural networks, clustering approaches, reinforcement learning and ensemble learning. The authors identify four major open research questions for future work:

i. continue with clustering-based approaches,
ii. additionally find explainable approaches,
iii. improve “slicing” methods to decompose the code, and
iv. the creation of better data sets.

Ciancaglini et al. [383] review the malicious use of ML from the attackers perspective. This work is a collaboration between Trend Micro, the United Nations Crime and Justice Research Institute, the Europol Cybercrime Centre, and the Centre for Artificial Intelligence and Robotics; it presents known malicious uses of ML, discusses potential future abuse scenarios, and provides recommendations on how to counteract. Finally, the authors include a case study on deep fakes with the attack vectors face re-enactment, face generation, speech synthesis and shallow fakes. In this case study, multiple cases are mentioned where deep fakes have been used, but the authors state the technology has not been used on a large scale as of yet. In contrast to the defenders tools, the authors state that most applications of machine learning for the attacker are still in their infancy. With respect to malware, for example, the authors conducted a large-scale search for malware samples and identified whether they had any references to machine learning libraries such as *TensorFlow*, *RapidMinder* or *PyTorch* with no result.

Shaukat et al. surveyed ML for cyber security for the decade of 2009 to 2019, reviewing related research of more than 300 publications [384]. The authors follow a similar categorization as Naik et al. [381], the identified cyber security categories are spam detection, malware detection and intrusion detection. The authors discuss each category in depth, showing which ML methods have been used, and also deail relevant sub categories such as specific attacks. Shaukat et al. also present relevant cyber security data sets and their use, and discuss relevant challenges centered around the respective ML models. They also discuss overarching challenges, for example adversarial attacks.

In the next sections, we follow the introduced categories in the surveyed literature and apply corresponding use cases for QML in the cyber security domain.

15 Spam Detection

The problem of email spam detection is both old and well-known. To a high degree, it can also be considered solved in the sense that many email products nowadays ship with advanced spam detection functionalities. Indeed, the topic has been approached using various ML techniques very early on, and is one of the few use cases for ML where both performing research and utilizing real-world applications have been done even before the turn of the century.

ML-based spam detection has been researched extensively and comprehensively. The topic is well understood in almost all its facets. This is clearly evident from an abundance of literature on classical ML for spam detection: there exist several systematic literature studies that are comprehensive, example the works of Hussain et al. [385], Crawford et al. [386] or Lota's and Hossain [387] examination of literature. It is also evident from the large selection of spam detection products (especially in the realm of email spam) that have both been continuously improved and perfected, and have long been accepted by the market as valuable means of filtering.

Furthermore, non-ML approaches have brought good results in many areas—for example, reputation-based and source-based scoring of email traffic is a usual augmentation of ML-based spam filtering in the area of email transmission. Similar concepts exist for other areas of spam, with the most dominant method being the request to the (browser-) user to pass a small test (Captcha: completely automated public Turing test to tell computers and humans apart), as well as simple to complex rate limiting.

Little to no research work exists investigating spam detection on QML. For many reasons, it is not to be assumed that practitioners seriously plan to build spam detection systems on quantum devices—neither in the near future, nor in the era beyond NISQ devices. Obvious reasons are both cost and complexity, which will likely never reach a positive return on investment given the current sophistication of spam detection in the classical space. In a similar fashion, a malicious actor will require an extreme motivation to invest the effort (and money) to utilize a quantum device in order to attack a spam detection system. Nonetheless, necessary tools and underlying concepts, such as text classifiers, have indeed been investigated in the quantum space and do receive constant or increasing attention.

In summary, the use case of spam detection represents an application of classical machine learning that, when transferred and translated into the space of QML, offers immense rewards when, instead of as a practical application, are understood as a research direction.

15.1 Definition and Current Research on Spam

In order to establish a consistent understanding of the term *spam*, we align ourselves with Kremling and Parker [388].

> "...spam is defined as unsolicited commercial electronic mail that includes any commercial emails addressed to a recipient with whom the sender has no existing business or personal relationship and not sent with the consent of the recipient, and commercial electronic mail is defined as any electronic mail message the primary purpose of which is commercial advertisement or promotion of products or service. "

They further explain that spam typically includes advertisements, business proposals, requests for favors, invitations to parttake in benefits and charity, as well as financial offers. Such mass communications are additionally used for more malicious intentions, such as phishing (impersonation with the aim of tricking a person to provide information, like banking credentials or other account data).

Thomas et al. [389] expand the concept of spam insofar as spam is also posted and distributed via services such as social networking platforms, product review, or in comments on blogs and microblogs, videos or text status update sites.

The practice is that scammers automatically generate accounts to spread spam and fraud through these services. According to the authors, spammers pursue not only fraudulent profit-driven activities, but also, for instance, disinformation or attacks to censor political statements.

Lau, Liao and Kwok [390] regard spam with reference to fake reviews. They analyze data sets from a popular e-commerce site. The authors explore the possibilities of using text mining for detecting online review spam. Here, it is enormously difficult to distinguish spam from ham, because fake reviews are written purposely to mislead readers, giving the impression of being true reviews. Moreover, they are often not generated by algorithms, but written by humans.

Hayati et al. [391] investigate the term "spam 2.0", and refer to so-called *spambots* specifically looking for vulnerabilities in the web to be able to carry out spam mischief in a next step.

A summary by Dada et al. [392] provides a comprehensive systematic literature research, including a summary of 21 papers on spam filtering using ML, together with the data sets used. The authors investigate the utilization of ML techniques for filtering spam emails by large providers such as Google, Yahoo and Microsoft. Moreover, they study the procedures of spam filtering on a higher level, including several approaches to spam filtering based on probabilistic methods, decision trees, artificial immune systems, support vector machines and artificial neural networks. Finally, the authors provide a comparative review of the existing ML approaches' pros and cons, as well as open problems in spam filtering. The authors conclude that deep leaning and deep adversarial learning are the leading methods employed for tackling spam email threats successfully in the years ahead.

15.2 The Underlying Approach: Text Classification

The field of spam detection is a concrete use case of text classification. A certain amount of substantial work and results are already available in this field of research. For instance, Shang et al. [393] address the k-nearest neighbors algorithm as it is broadly applied for text classification. They highlight that the number of features contained in text reaches the scale of several thousands, which makes similarity calculation computationally expensive. Since a majority of classical ML techniques can hardly handle a large quantity of vectors in highly dimensional space, the authors see an advantage in QML algorithms: they are very well able to handle high-dimensional vectors in huge tensor product spaces and exhibit exponential speedup over their classical equivalents. This gives QML a serious advantage over classical machine learning in the field of spam detection. It should be mentioned that Wiebe, Kapoor and Svore [196] did not observe such (an exponential) speedup by QML against classical ML in classification tasks using k-nearest neighbors.

Liu, Yang, and Jiang [394] introduce a new quantum-based supervised learning approach for text classification. The novelty of the approach is that it performs classification in terms of a process of a physical system. That is, the classifier makes use of the fundamental equation of quantum mechanics, namely the *Schrödinger equation*. The authors validate their method using the Reuters-21578 text classification collection data set[7], as well as oral conversation data sets. Specifically, they compare the performance (in terms of accuracy) against classical SVMs and k-nearest neighbors. As a result, the quantum-based classifier exhibits a good performance in text classification and can compete with both classical variants. In addition, the SVM and k-NN each have their quantum counterparts, with implementation tutorials being widely available online.

At the same time, the authors observe that their method performs not always optimally when it has been fed with a large training data set, whereas it outperforms the classical SVM and k-NN on small-scale training sets.

These observations point to an apparently common phenomenon across many use cases: QML does not outperform classical ML on large training data sets—better said, QML might show even lower performance. Conversely, however, QML seems stronger on smaller data sets.

[7] https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection

> This does not only seem to hold for text-related tasks, as Reese et al. [369] observe this behavior in a quantum system for vision-based industrial quality control. This effect is considered in more depth in section 18.4.

Shi et al. [395] develop an “interpretable complex-valued word embedding” method (ICWE) for designing two quantum-inspired deep neural networks: the ICWE-QNN, and the CICWE-QNN (convolutional complex-valued neural network, which is an extension of ICWE). These methods perform binary classification of texts, spam detection is a use case for them. The authors use the predominance of deep learning techniques like Gated Recurrent Units, self-attention, and CNNs to extract text features. They demonstrate that deep learning computations can be integrated into quantum-inspired complex neural networks. The authors use five binary classification data sets to evaluate the performance of their two methods against existing traditional methods and against the existing quantum-inspired methods. The existing methods are known to be highly accurate, and the CICWE-QNN shows very good performance in comparison, even outperforming the traditional methods on several of the data sets under consideration.

Baronia [396] develops a proof-of-concept for classifying texts, utilizing four variants of a hybrid quantum-classical model. A variational quantum circuit layer with the same number of inputs was added to each model. One model showed a measurable improvement in accuracy compared to its classical counterpart. The author observes that it requires considerably more time (measured in minutes per epoch for a given batch size) to train the quantum models versus the classical ones. He also mentions that with the foreseeable development of powerful quantum systems, QML can offer real advantages in terms of efficiency. He further notes that currently, if quantum speedup in a hybrid model is the goal, then one can most effectively achieve it by transfer learning for a downstream task. The author thus concludes that, in the near term, it is likely that research will lead to more effective usage of quantum circuits in such models.

Arthur and Date [397] take the physical performance limitations of classical ML as motivation to develop a neural network architecture for binary classification. The authors validate their architecture on simulated hardware and state that their hybrid approach classifies 10% more accurately and performs 20% more efficiently than an individual variational quantum circuit. They notice that models score well on real quantum devices as long as the number of qubits and gates is kept low.

16 Malware Detection

16.1 Historical Signature-Based Approaches

A historical approach for the detection of malware is signature-based detection: the cryptographic hash of an executable is simply compared to known bad signatures. This method has two obvious limitations. On one hand, it can only detect malicious files that have been seen before, have shown malicious behavior, and have been added to the detection database. This updated database in turn needs to be available at the time of signature comparison. On the other hand, the detection database needs to hold an ever-increasing amount of signatures.

In the arms race between attackers and defenders, this battle was lost to the attackers: in today's world, attackers mostly have the upper hand in signature-based detection. Tools such as morphing and self-encrypting code are at the attackers' disposal, substantially increasing the number of signatures for malicious files. Furthermore, today's networking and computing resources allow for each target to be attacked with an individual, "use-once" malware binary.

As a side note, a radically different concept to respond to the same issue of preventing execution of malicious code is whitelisting. Here, only specifically selected, examined, and trusted binaries are included in an allow-list. Deep within the operating system, any requested execution is checked against this list. This concept obviously comes with separate advantages and drawbacks.

16.2 ML-Based Approaches

The deployment of ML has proven to be a valuable method to counter the attackers' strategies above. One of the most important aspects, as with any ML deployment, is the selection and extraction of features. For the analysis of binaries, two different strategies have emerged: the *static* analysis, which is based on features of the binary file itself, and the *dynamic* analysis, which is based on the behavior of the executable during runtime. Since this analysis relies on the execution of the code in question, it presents itself with its own challenges. Two aspects illustrate these: the requirement of a specialized, secured environment ("sandbox"), as well as time requirements. On the other hand, however, this analysis method allows the extraction of features that are otherwise unavailable, and at the same time manages the extraction of threat and behaviour indicators that are much more difficult to hide by the creator of malicious code.

In figure 11, examples for features in both strategies are compared. Of course, features based on both strategies can be used, and it even proves worthwhile to employ a two-stage analysis to increase detection accuracy while conserving resources.

Static Feature	Dynamic Feature	Details
strings		sequences of n or more letters
entropy measurements		characterizing compressed, encrypted, or delaying code
internal function calls	function call trace	use of internal functions, procedures and subroutines
control flow and subroutines	instruction trace	detailed mapping of jumps, calls, and loops
API function calls	API trace	mapping of used operating system functions
specific network function calls	network trace	patterns of network communication/information flow
opcode tokens	register use	use of CPU instructions and parameters
	memory use	use and layout of memory

Figure 11: Features in malware analysis

Quantum computers excel with graphs: The problem of malware detection is well-suited for a description using graphs—the translation of this classification into equivalent expressions of graph problems can involve, for example, graph similarity or graph matching.

The analysis and computation of graph-related problems is an area where quantum computing can show huge advantages [398], and perhaps even superiority over classical systems [399].

The combination of QML with graph theory has the potential for even further increases in speed and accuracy—both inside and outside cyber security.

Quantum computers excel with approximation: In a similar fashion, a large subdomain of malware detection is the search for similarities in structure or behaviour. Using algorithms based on probabilistic and approximate comparisons, a malware classifier can be constructed. This can be understood as a simple extension of the universal approximation theorem, which is also applicable in the quantum world [400]. A significant approach are variational quantum eigensolvers, and therefore quadratic binary optimisation, as we have seen in chapter 5.3.10.

Further research into the properties of resulting classifiers might similarly show advantages or perhaps superiority of this class of techniques in the quantum space.

16.3 QML Approaches and Current Research

In their extensive review on the subject, Gibert et al. provide a complete list of methods, and a comprehensive explanation of features used for malware classification [401]. They focus on deep learning techniques, and provide insights on utilizing the advances of computer vision by transferring code into images. Additionally, they show limitations of the approach as well as current trends, and include an overview of the current research and research challenges.

One of the results of Gibert et al. is the correlation of feature selection of the binary with the respective algorithms used for classification, at least to some degree. Potentially, this is merely a symptom of research still branching out in different directions, and therefore an indication that the research in this direction has not yet reached a sufficient level of maturity.

However, an additional interpretation is that various approaches are suited for detecting unwanted code, meaning that the selection of the ML algorithm is less important when compared to other aspects, like the selection of features. However, especially when viewing the task as a binary classification into the categories of benign and malicious software, the choice of the ML algorithm can have a major impact on the accuracy for samples that have a high degree of dissimilarity with training data.

A similar (more generalized) investigation of the correlation of features with algorithms in the QML world might provide deeper insights—and ultimately could shed additional light on this open question even in the classical world.

As the selection and extraction—perhaps the "engineering"—of features for malware detection clearly carries major importance, a worthwhile examination is the move of this aspect into the ML model itself. The features are no longer determined beforehand or remain static, but instead become an early stage of the trained ML model. Important examples for this strategy are the major improvements observed in the applications of deep neural networks. This method has proven to be a revolutionary idea in the field of computer vision, for example, enabling a sudden and significant improvement in classification power over previous generations.

The approach naturally exhibits some drawbacks: mainly, the requirement for a much higher number of samples, as well as the surge in model complexity. Deep neural networks become deeper. Moreover, especially on NISQ devices, the circuit depth, closely related to the model depth, remains extremely limited.

Utilizing advances in image recognition can be an option whenever it is possible to translate the classification subject into some kind of image. In the case of malware analysis, an image can be crafted out of the binary, for example by converting bytes into 256 shades of gray, and thereby pixels. Metadata can be added in a similar fashion, and encoded as additional pixels. After these simple preparatory steps, the tools of image analysis are available for malware detection and models can be trained to classify binaries. It is noteworthy that specialized hardware for image classification neural networks provides significant computing power, which is harnessed by following this idea. Naturally, similar specialized hardware is available for various other classification tasks in the space of classical ML. Due to specific optimizations, such computing power is not available in this form for most other purposes, especially for malware detection.

However, taking away stages such as tokenization of the code and other feature extraction and pre-processing aspects, has the obvious drawback that the computer vision model needs to incorporate the extraction of this knowledge within the model itself. Furthermore, different activation functions are needed for this "hybrid" application.

Incorporating existing QML infrastructure of quantum circuits, knowledge, and research results into the questions of classification between malicious and benign binaries might be a promising path—especially at the outset of this new research area.

Altogether, it remains to be proven that this concept has advantages over more thorough approaches, or at least can add insights.

On the other hand, viewing the concept on a more general level, one can find that the strategy of transforming one classification task into another, preferably one that is studied deeper and implemented more efficiently, has potential to boost initial results in QML research.

(Partially) encrypted or compressed binaries will certainly remain to be an unsolved problem. The notion of deprecating any encryption is an absolutely bad idea. This fundamental truth applies to binaries and code in the same way that it applies to other (digital) data. In the environment of cloud providers, containers, and autonomously acting agents, as well as the presence of attackers and attack vectors throughout all of these environments, there are good reasons to utilize encryption.

If the training set includes malware samples that employ encryption or compression, but no such benign samples, the classifier will quickly learn to score these properties negatively—effectively removing the option for future benign binaries to employ the same methods. This would imply more than a deprecation of encryption, it would almost entail outlawing it. It is also noteworthy that, as long as the feature selection does not include measures to "look into" encrypted or compressed data, any other feature will essentially disappear.

Thus, care must be taken to ensure the presence of decryption and decompression routines, as well as sections of data with high entropy, do not become over-weighted decision factors for the classifier.

In conclusion, a tokenization of the code in question, along with other thought-through feature extraction methods, is much more suited for continuing current research.

> An approach with a comparatively small set of well-selected features could be greatly rewarded by quantum speed-ups and thus might develop into a valuable research direction. Moreover, because of the difficulties with data encoding, a feature-extraction phase might not only make the classification using QML possible, but could turn out to be where vast speed-ups might be gained.

It is clear that research is in its infancy even in the classical space. While the problem of spam detection has seen significant advances some years ago, this development is only now finding traction in the detection of malicious binaries. The much higher requirements for efficient and accurate detection are met only recently—requirements such as advances in computing power and cost, or the evolution of the field of ML itself. The availability of proper and plentiful training data remains to be a limiting factor for both research and product development, and a difficult issue to solve. The choices for public benchmark and training data sets are extremely limited. As we will recognize in section 18.2, the aging of samples is not only an important factor, but also reaches high rates in the current cyber security environment.

In essence, a continuous flow of labelled samples is needed, which implies a high degree of automation. The concept of reinforcement learning is able to alleviate issues like this—although it comes with other drawbacks. For example, reinforcement learning is very perceptive to adversaries poisoning the training data. Therefore, methods are needed to ensure only "valid" samples and labels are processed. Considerable efforts need to be spent to avoid attackers' data being included in the wrong way. The identification of malicious input thus becomes somewhat of a recursive problem. However, as there are indications that QML could be more robust than a classical model given small amounts of training data (please refer to section 18.4), there might be good reasons to extend research and eventually employ QML for the classification of binaries into benign and malware categories.

Classifying binaries is undergoing a shift towards ML right now, as evidenced by the fact that while ML is used for malware detection in production environments, it is only one indicator of many. ML will likely receive further attention, become more common, and grow increasingly accurate in the next decade. It will necessarily augment, and potentially even phase out, most signature-based (and other) approaches.

As we have seen, various bottlenecks restrain and slow down the development of further research, especially in the QML space. In certain aspects, QML research might prove to offer a deeper understanding, at least through supplementary observations.

16.4 Detection of Malware in Quantum Computers

As any system can be the target of an attack, the detection of malicious modifications of quantum computers will eventually become a necessity. Providers of quantum computing power will disallow certain actions and thus perform certain checks on the input (data, circuits, and perhaps additional configuration) before execution. Various obfuscation methods might be employed to circumvent these measures. This might go as far as exploiting certain properties of the quantum system, especially as long as attackers can select specific systems for their activities.

Frequent checks are performed on quantum computers to ensure stability, report on properties like error rates, as well as to detect failure indicators. A countermeasure against several attack types can be incorporated into these checks: using known-good input and output, a sanity check can show malicious misconfigurations of quantum circuits or the quantum computer as a whole.

The methods described in section 17.4, specifically the ideas derived from [402], might be employed for this purpose.

QML could be employed to detect methods like obfuscation, encryption, and even exploitation patterns when analyzing the data and code input to a quantum system to secure it. Performing these kinds of QML checks on quantum circuits and eventually even quantum data is certainly some time away, and would only be seen beyond the NISQ era. However, less and less hybrid devices (and pure classical computing) will be used for certain computational purposes. Therefore, such analysis will become a necessity for continued benevolent operation of quantum devices.

With the fast development of quantum computers, further research has to be done to determine possible means of misuse, as well as corresponding approaches for detection and prevention.

Malware might not be relevant within the next few generations of quantum computers and perhaps not within the entirety of the NISQ-era. Nevertheless, it will certainly become reality at some point.

17 Attack Detection

The field of attack detection includes multiple subareas and -aspects. The most important of them is intrusion detection, which can be performed in the network, on endpoints, or in centralized logging environments. It also includes behaviour-based intrusion detection, which builds on the behaviour of systems and users.

In the current research literature, QML is not thoroughly examined for the use of attack detection. Mainly, the context of the detection of network intrusions is regarded. For this reason, this section focuses on a few specific areas.

17.1 ML for Attack Detection

Lifandali and Abghour [403] take as motivation the importance of the fast growth of computing networks for their investigation on how to protect networks using deep learning techniques. The authors speak of a so-called "data-driven intelligent intrusion detection system" that utilizes machine learning to detect intrusions and anomalies in networks. The study explores the potential of deep learning techniques and provides a literature review on the performance and limitations of ML-based intrusion detection techniques. They inspect limits as well as assessments of the models. They additionally include surveys of the benchmark data sets that have been used to train models.

17.2 QML Approaches and Results

Gouveia and Correia [404] explore SVMs for intrusion detection based on network traffic classification, and make comparisons between classical and quantum implementations. They validate their model through simulations and demonstrate that the accuracy can compete with classical SVMs. They use a quantum computer for the application of the model, while feature learning is done classically (which they call "quantum assisted computation"). While results were similar between the classical and quantum approach, measuring the accuracy of the QML models in their simulation points towards a certain advantage for the QML side.

Suryotrisongko and Musashi [405] investigate several hybrid quantum-classical approaches to botnet detecting domain generation algorithms (DGA). A classical fully connected CNN deep learning model was used as a benchmark, and compared with a quantum CNN model. They employed their own DGA data set. They explicitly added noise during their experiments, as they were based on simulations. The authors found that the initial randomly seeded values have a considerable effect on the performance of the comparisons, and thus move to setting identical initializations. The best results were obtained from fully entangled encodings. These also take more time to compute, leading to potential bottlenecks. As the volume of input data increases, model performance between the two stabilizes. On very voluminous data, classical models perform better. Overall, it can be concluded that the approach of a “simple addition of a quantum layer” does not immediately provide easy advantages.

Payares and Martinez-Santos [406] provide a look at the use of QML for detecting DDoS attacks. They work with three classification approaches: an SVM, a neural network, and an ensemble model. They base their research on the DDoS Evaluation Dataset[8]. Their results indicate a considerable improvement in accuracy over classical ML. They also go the important step of publicising the quantum code they use. However, the authors admit that their work only represents a very small first step in detecting threats like DDoS attacks.

17.3 Advanced Detection

Zoppi et al. [407] address the problem of zero-day attacks using classical ML—the exploitation of vulnerabilities that are unknown to the defender. Unsupervised anomaly detection seems promising for zero-day detection. However, since detection performance is not ideal when unsupervised algorithms are used as the main tool, a synergy between supervised and unsupervised algorithms is likely to emerge. Using a public data set, the authors show how to develop an unsupervised anomaly detection algorithm that builds on meta-learning to significantly improve detection performance and, in particular, increase robustness to zero-days.

Deep reinforcement learning (DRL) techniques are known to be reactive, adaptive and scalable—characteristics that are desirable for the detection of today’s complex and dynamic attacks. Nguyen and Reddi [408] review the DRL methods that are specifically designed for cyber security in the classical space. They address different areas of DRL-based protection, including cyber-physical systems, autonomous intrusion detection, and multi-agent game-theoretic simulations. It should be noted that the authors do not address the use of QML or quantum computing.

> Quantum Distance Based Classifiers (QDBCs) can be considered a novel approach in QML that is not based in classical ML. QDBCs are similar to k-nearest neighbour algorithms, and were introduced by Schuld, Fingerhuth, and Petruccione in [201], and extended by Blank et al. in [409].
>
> They could be a viable method in attack detection, as the method has certain intricate advantages, like the applicability of kernels, and an inner structural simplicity. However, disadvantages include a high resource need—in certain circumstances, the number of qubits needed is linear in the number of data points and features.

[8] `https://www.unb.ca/cic/datasets/ddos-2019.html`

17.4 Quantum Algorithms for Quantum Data

The detection of anomalies can be considered an important step in the detection of an attack. Liu and Rebentrost [402] investigate QML for the detection of anomalies in quantum data. Their aim is to provide processing means for "big quantum data". As they state, for some applications, this might eventually become an important concept. Two classical anomaly detection approaches are kernel principal component analysis and single class SVM. The authors present quantum counterparts for both to detect quantum state anomalies. They discuss the complexity of the approach, asserting very high complexity reductions.

Their approaches are fairly broad and certainly not specific to detection of the "presence" of an attacker—after all, outliers, irregularities and exceptions are used in many different kinds of data analysis today, and will analogously be used in the examination of quantum data.

Nonetheless, as soon as big quantum data is a more common concept, and quantum analysis of this data is more commonplace, it will become important to possess the tools necessary to detect, for example, maliciously inserted data points, or malicious modifications.

18 Overarching Considerations

18.1 Measuring, Comparing, and Optimizing (Q)ML Models

Measuring the performance of machine learning models is a non-trivial problem. While various measures have been proposed, research is very much ongoing. An insightful comparison between different approaches relies on suitable measurement metrics and methods.

Abbas et al. specifically investigate the question of measuring generalization performance (the ability of a model to classify previously unseen data points) [410]. This topic is especially relevant in the application of ML for cyber security, where attacks frequently involve completely novel and previously unseen methods, approaches, and patterns. Even worse, there are comparatively high requirements of efforts for the generation of training data when ML shall be applied for cyber security. With the ever-ongoing evolution both of technology used and attack concepts, the training data requirements will stay an open question for some time—and so will the need for generalization performance measurements and comparison.

Various approaches exist for measuring the performance of ML models. Depending on the insights desired, different measurements make sense. Essentially, all of these approaches are applicable to QML models as well. Nonetheless, formulation of new methods specifically suited to measure the properties of QML systems, should be studied.

Additionally, there are currently no consistent standards for comparisons between classical ML and QML. More exact measurements of the similarities of and differences between the two has the potential to uncover currently unknown properties of QML systems.

While employing NISQ devices, various limits are imposed on QML algorithms and models, as we have discussed in section 9.2. Additionally, various trade-offs exist—employing more complicated models with higher circuit depth or more qubit requirements, for example, has the implication that decoherence destroys the results of the computation before completion more often. Especially in neural networks, there is a trade-off and direct correlation between network depth and quantum circuit depth. A straightforward consequence is that any optimization can produce big differences in outcome. In many applications, a highly beneficial area for optimization is the selection of features: a reduction of the feature space has great potential for reducing the requirements regarding data encoding and amount of qubits.

18.2 Model Decay

Another important aspect has been highlighted already: the need for training data. ML-based detection has inherent limitations with regard to unseen data, depending heavily on the amount of similarities. Within the setting of highly motivated adversaries, combined with today's requirements for accurate and real-time detection, a gradual or even sudden shift of attack classes and strategies can leave an ML model outdated quickly. However, the consistent development of novel attack and intrusion methods with new properties, techniques and features constantly introduces additional sources of such dissimilarities, and therefore causes a divergence of the used training data to the classification input.

This phenomenon is captured in the term *concept drift*. Concept drift occurs where a static relationship between observable phenomena and corresponding predicted targets is gradually disappearing. As a result, as new attack and intrusion methods are introduced, prediction accuracy and thus the reliability of the system declines.

> Much research regarding cyber defense happens behind closed doors. As long as there is a market for security products, there is competition, and therefore trade secrets. In most areas, public data sets for deep research are scarce: existing data sets are comparatively small and mostly very old. They do not cover newer types of attack, and most originate from simulated environments. One noteworthy exception are malware samples: fairly large sets of samples and data points can be acquired for research and development.
>
> Enabling further research by developing and publicising new data sets is necessary.

A frequent update of the detection models can counter this important issue, both in classical ML and QML. Another solution is the introduction of reinforcement-based learning approaches. Here, the classification input and output is immediately used for updating and incrementally re-training the model. This also connects to an additional strategy for the defender: the classification should not simply have a binary output ("malicious" or "benign"), but instead output probabilities for multiple classes or types of attacks or malware—and correspondingly of non-malicious categories.

18.3 Synthetic Training Data

In most cyber security data sets, the number of non-malicious samples significantly exceeds the number of attack samples. The quality of an ML model is heavily dependent on the composure of the training data set—meaningful ML training requires a fairly balanced data set. A Restricted Boltzmann Machine (RBM) can address this by modelling the underlying probability distribution of the data set. In this way, synthetic data can be obtained from the RBM. This is then used to balance the data set.

It is noteworthy that this approach differs from other synthetic data generation methods such as “Smote” [411], which generate data that does not necessarily follow the underlying distribution of the data. Another alternative is employing Markov chains for the creation of synthetic data, as described for example by Chalé and Basian [412]. The use of Markov chains is computationally complex and resource intensive.

Dixit et al. [300] train an RBM a with a quantum annealer (QA), using the imbalanced Cybersecurity dataset ISCX 2012[9]. They show that after this training, the annealer is able to output synthetic training data of high quality. They also compare the use of quantum annealing and the classical method of contrastive divergence for training an RBM and found that the generation of synthetic data was equally good with both methods. The QA-based training was significantly faster and can be further improved with the number of available qubits increasing. Ultimately, the authors find that application of these techniques has the potential to provide a benefit to the accuracy of intrusion detection applications. Strictly speaking, this improvement, while based on the use of quantum technology, can even be applied to classical machine learning.

18.4 Robustness and Expressability in QML

A frequent crucial issue for machine learning research and development is the lack of large amounts of high quality and balanced training data. These two often negatively impact the generalization capabilities of correspondingly trained classical ML models. Dunjko, Taylor, and Briegel investigate the learning efficiency of QML systems [413]. They base their work on a very formalized approach of a general agent-environment framework, and build a schema to investigate improvements in QML. As a result, they state that quadratic improvements can be observed.

In a noteworthy contribution, Abbas et al. [309] thoroughly examine quantum neural networks with regard to their generalization bounds and expressability. They use information geometry, especially the Fischer information spectrum. They show that, under the right circumstances, QML models can have a substantially higher effective dimension than classical ones. The important outcome is that QNNs show advantages with respect to the required training effort which the authors attribute to their ability to reduce non-optimal conditions for the optimization, namely barren plateaus. These have been discussed further in section 9.6. They emphasize that QML is able to demonstrate advantages over ML in this regard. These advantages also translate to a more robust learning process. Most importantly, they practically demonstrate their claims using a 27-qubit quantum computer in a simple ML model (the depth of the feature map is only 2).

[9] `https://www.unb.ca/cic/datasets/ids.html`

> Whether or not this training advantage of QML models holds in general clearly is still a contested question and needs further research. To emphasize, especially for applications in (defensive) cyber security, further research in this area has the potential to considerably advance cyber security ML—due to the continuous changes in cyber security systems, as well as due to the arms race between attacker and defender, adequate training data in required and desired quantities will remain rare.

A frequently mentioned hope with regard to benefits of QML is that quantum encodings of data could implicitly represent higher-order features that are impossible to encode classically. Thus, due to the major role entanglement plays, the data encoding itself may provide a quantum advantage for classification.

A multitude of aspects follow. Among them is the notion that a QML system might possess more expressability and robustness in comparison to a classical ML system—given the same (amount of) training data, the QML model would be able to, in a way, “extract more knowledge” than the same classical model could. In other words, a QML system would require less input samples than a comparable classical system to achieve the same prediction capabilities.

> Indeed, a recent contribution [414] suggests that the possibility of entangled data representations is a unique benefit of QML. However, while entanglement may potentially allow for encoding (non-classical) dependencies, there may be trade-offs with regard to required circuit depths. Further research will be needed to establish and examine realistic advantages in situations of scarce, imbalanced data.

18.5 Adversarial Attacks in Cyber Security

For the adversary, detection usually constitutes fairly catastrophic results. Any detection means that they must at least start all over again, and, more importantly, they risk alerting the target. Therefore, there is a lot of motivation to carry out an attack that is classified as benign by the respective attack detection model (be it malware classification, attack detection, or any other threat discovery). The attacker is thus deeply incentivised to employ adversarial methods as described in chapter 11.2. A comprehensive review of adversarial deep learning is also done in [319].

Taking malware detection as an example for such a generation of adversarial input, the starting point can be a truly benign sample. Numerous small, almost insignificant changes are then tested against the classifier. These are usually called *perturbations*. Using an evolution approach, the best scoring samples are used as parents for the respective new generation of samples. Approaches comparable to evolution also include, for example, gradient descents and generative adversarial networks (GANs). Following this path, the samples gradually approach the functionality the attacker aims for, while ensuring the classifier remains to be fooled. Similarly, the starting point can be an easily detected malware, which is gradually changed into a binary classified as benign by generations of minimal modifications that do not change the original functionality.

The generation of adversarial samples usually relies on the attacker having full access to the targeted model. However, this need not always be the case—sophisticated attacks can be carried out that circumvent this limitation.

One of the major outcomes of Biggio's research [313] is that in the space of cyber security, ML is vulnerable to adversarial attacks. However, these specific cases of adversarial attacks have not yet received much research attention in the quantum space.

> It therefore remains an interesting question not *whether*, but *how much* the creation of adversarial samples, whether for an ML or a QML target, can benefit from an implementation in QML. It can be assumed that the classification tries of each new adversarial generation can heavily benefit from quantum encoding and processing: a large number of tries can be classified in parallel and amplitude amplification may then be used to select only those results with the highest score as parents for the next generation.
>
> Especially if both the generation algorithm and the attacked classification model are combined within one circuit, adversarial examples might be created almost instantaneously and at the same time come with a very high (and incorrect) confidence of the attacked model.
>
> One drawback deserves attention, however. Even if the computation itself is fast, the issue of data loading and encoding might prove to stand against quadratic (or higher) speed-ups—as it does in many other applications.
>
> There is also no generalized algorithm that takes an ML model as input and generates an equivalent QML model: Creating a (fairly exact) "copy" has not been achieved yet in terms of a constructive method. Attacking an ML system using QML requires a fairly exact quantum duplicate.
>
> Some ideas and questions posed by Goodfellow, Shlens, and Szegedy in [320] show even deeper aspects of this interesting research area, among other things they investigate reasons for the broad vulnerability of models, and discuss counter measures including ML algorithms that are much less affected.

18.6 A Case for Deep Learning in Cyber Security QML

The phase of engineering, testing, and optimizing a good feature set is an important step during the creation of any ML model. It not only enables, but also vastly improves (or deteriorates) the accuracy of the resulting classifier. Therefore, this task requires knowledge, experience, and considerable resources. Of course, a property of the resulting model is the evaluation and weighing of the features while it processes the feature vectors into the classification outcome.

As we have seen already in section 18.2, the area of cyber security imposes specific needs upon any kind of detection and defense: a high capacity for adaption to new risks coupled with the ability to identify threats that lie well outside of previously seen attacks. That means that especially for cyber security, close attention should be paid to any improvement regarding these needs.

Deep neural networks have a crucial advantage in this aspect: the selection of features is moved into the model itself. That means that the resulting DNN *implicitly* finds and selects features. The cost incurred is the added complexity within the model, as well as the need for a higher amount of training samples.

> Because of the inherent properties of quantum computing, implementations in QML might exhibit these advantages even more dominantly. This is a prime candidate for research into where specific advantages of QML over classical ML might lie.

It should additionally be mentioned that the techniques used in reinforcement learning can provide additional benefits to the continuous detection accuracy of resulting models. As always, these benefits come at some cost, for example, additional attack surfaces and an overall more complex model.

18.7 An Attacker with Dominant Quantum Capabilities

Perhaps one of the most important questions in the context of cyber security is: What are the implications of a scenario where a malicious global player (a nation-state or a corporation) possesses dominant quantum capabilities?

Because of the potential for high reward, a part of quantum computing research and engineering is done behind closed doors. It can be expected that certain advances in quantum computing power will not be generally available or even known to exist. The big players in cloud computing outclass the research and IT resources of most countries, so a common view is that it would be somewhat unlikely that a significant non-public quantum advantage might exist eventually. However, for various reasons, a malicious actor might find ways to both achieve a major quantum advantage, and hold it secret.

Disregarding the likelihood of this scenario, for the purpose of this section we will assume that a malicious actor has “sufficient” quantum capabilities in gate count, circuit size, and depth, has eliminated noise and achieved long quantum coherence times, and is able to utilize extensive numbers of fully error-corrected qubits, and, finally, has mastered data encode along with input to and output from the quantum system. In short, the hypothetical malicious actor has left the NISQ-era and fully entered the quantum era.

Naturally, the attention would first be directed at crypto-analysis as this malicious actor could obliterate public-key encryption and easily read encrypted data. This is the motivation for NIST’s efforts to find public-key cryptography algorithms that remain secure in this scenario[10].

As we have seen in section 18.5, using QML, it is reasonable to assume that adversarial examples against classical ML models can be created quicker, more plentiful, and with higher quality and non-detection properties. Where other, classical attackers require more knowledge about the attacked system, the quantum-enabled malicious actor can potentially choose attacks requiring less knowledge based on the (in effect) higher computational power and QML capacity of the quantum devices. A similar argument can be made for the number of required interactions with the system, and thus the probability of detection of the attack.

In the domain of malware detection, the malicious actor might thus be able to generate exploits using QML that completely evade detection. Correspondingly, they might be able to more easily find vulnerabilities and exploitable bugs, and then use these as a basis for improved malware creation. Similarly, the malicious actor might be slowed down much less by intrusion detection systems and other means of attack detection, having powerful tools for evasion.

Altogether, they will have much more freedom to gain initial access and move around in attacked IT systems, as well as significantly improved means to exfiltrate data (attacking confidentiality), to shut down or destroy systems (attacking availability), and to quietly modify data (attacking integrity).

> The research into post-quantum cryptography is motivated by the notion that this kind of worst-case scenario, regardless of it’s likelihood represents a possibility within the next few decades. The move for standardization of new cryptographic primitives is one defensive measure against the effects of this scenario.

[10] https://csrc.nist.gov/projects/post-quantum-cryptography

Similarly to this post-quantum cryptography push, the research into wider defenses for both ML and QML systems against this malicious player might be considered crucial to the continued stability of the digital world. While representing only one measure of many to oppose the attacker, the development, establishment, and standardization of suitable protective means can be regarded as appropriate.

Moreover, most efforts in this direction will lead to a general improvement of IT security, ML systems, as well as QML systems, and thus should not hinge on the likelihood of the worst-case scenario. Simply phrased, the research into and consequential improvement of security within the cyber space always comes with a reward—even though measuring the return on investment often is extremely difficult and usually requires taking into account fairly long timeframes.

19 Conclusion and Outlook of Part III

Since quantum computers have become a technical reality and since it seems that their capabilities will continue to develop rapidly, experts expect that quantum advantages for certain kinds of (demanding) computations can soon be harnessed in practice. For the booming field of machine learning this suggests that quantum computing technologies and algorithms may find realistic applications at various stages of the machine learning pipeline and may thus lead to more efficient or faster training procedures than currently possible.

This, in turn, raises questions as to which application areas might be disrupted by QML. Here, an area that immediately comes to mind is the broad area of information- and cyber security not at least because one of the early success stories of quantum computing research was Shor's realization that universal quantum computers might easily break public key cryptography [36]. While his insight is disquieting, it also gave rise to long-standing research efforts on post-quantum cryptography. However, IT- or cyber security involves more aspects than just secure communication, and it is in this more general context where applied machine learning may lead to increased risks or improved security, alike. However, as of this writing, it is important to note that while modern learning systems can solve a wide variety of demanding cognitive tasks, their current human-like cognitive capabilities are a relatively recent phenomenon. It is therefore again just recent that ML solutions find increasing application as offensive or defensive tools in cyber security and that a post-quantum point of view on this domain is still largely missing.

In order to fathom the potential role of (future) QML solutions in cyber security, we therefore briefly reviewed protection goals for IT systems and how these are currently impacted by classical ML solutions. Based on this preparatory review, we then analyzed to what extent QML might change the current situation.

Given the technical capabilities of present day quantum computing systems, our analysis necessarily involved a foresight perspective. Our systematic survey of the existing literature as well as our extrapolations into the future therefore led to more open questions than conclusively answered ones. Put differently, as of this writing, the use of QML solutions as tools for attackers and defenders in cyber security scenarios is still largely a white spot on the research map and there are many directions for systematic future investigations. To be specific, the following list contains promising research questions for the near- to mid-term future that may help to better assess the role of QML in cyber security:

Is quantum feature selection superior to classical feature selection? Machine learning applications such as spam- or malware detection crucially depend on the quality of representations of the data that are to be analyzed. In other words, it is often not so much the algorithm but the (data) features that make or break the success of a learning solution. In this regard, it is thought that especially quantum entanglement could lead to better representations. However, the interplay between QML algorithms and quantum feature representations needs to be investigated more carefully than has been done up to now.

Does QML have a clear generalization advantage over classical approaches? Appropriate training for security applications is often scarce so that there are efforts towards generating or synthesizing useful training examples. This requires particularly robust generalization capabilities of a trained model and initial research suggests that, "under the right circumstances", quantum neural networks can generalize better than their classical counterparts. However, this training advantage in QML clearly is still very much a contested question. From the point of view of (defensive) cyber security, more research is clearly warranted because —due to the continuous changes in cyber security capabilities and the arms race between attackers and defenders—training data for defensive cyber security systems will remain rare for the foreseeable future.

Where could QML really shine in cyber security? In addition to potentially better dealing with scarce training data, what are other bottlenecks of classical machine learning that could be improved via QML solutions? For instance, quantum computing allows for efficient approximate linear algebraic computations; here, approximate comparisons between the structures of known malware and benign samples might be a worthwhile avenue of quantum classifier research. Moreover, by the very nature, quantum computers can excel at combinatorial or discrete optimization problems. From the point of view of machine learning, this suggests graph-based learning models as natural candidates for QML solutions. It is therefore interesting to note that problems such as, say, malware detection are well suited for graph-based modeling. Further research into the direction of translating malware detection problems into graph problems appears to be auspicious.

Could QML improve adversarial learning? Again, pertaining to machine learning for generative modeling, adversarial learning has become a predominant classical paradigm. Here, it seems reasonable to assume that adversarial example generation can benefit from quantum encoding and quantum processing since numerous tries can be classified simultaneously and amplitude amplification might allow for selecting candidates for the next generation. Especially, if both the generation algorithm and the attacked classification model are combined within one circuit, adversarial examples might be created almost instantaneously and, at the same time, seem to come with a very high (incorrect) confidence of the attacked model. Whether or not these expectations are really justified, however, needs further research.

All in all, at this point in time and given the current state of the art, the question of whether QML will positively or negatively impact the field of cyber security can not yet be answered definitively. Instead, the cyber security community will likely have to wait for significant progress in (applied) quantum computing and to carefully monitor ongoing developments before drawing authoritative conclusions.

An additional complication further hampering definitive answers to the above question lies in the likely assumption that ongoing research and development on quantum computing and its applications may not necessarily be publicly visible. As cyber crime and cyber warfare are substantial threat scenarios in the 21st century and as there are players who see capability in these areas as strategic assets, one cannot exclude the thought of financially strong, potentially even state-sponsored organizations which realize potential advantages of quantum computing and QML and are secretly pushing required technologies. Against this backdrop, it appears likely that, once available in a scalable and robust manner, quantum computing and QML will initially shift the imbalance further towards attackers. For the time being, it merely remains a plausible assumption (which still needs to be supported by further research and empirical evidence) that QML systems will serve as a means for cyber attacks rather than for cyber defense.

However, the current uncertainties regarding any further progress in quantum hardware and the large scale applicability of quantum machine learning also offers opportunities. In particular, since quantum computing is still a nascent field with a recognizable yet still not realized disruptive potential, IT security researchers, developers, and policy makers can "stay ahead of the curve". This is to say that it appears reasonable and appropriate to, first of all, stimulate and increase research efforts on QML defenses as well as to, second of all, rally all relevant stakeholders to begin to fathom necessary and appropriate standardization measures or guidelines.

Literature

[1] G. Hinton et al. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”. In: *IEEE Signal Processing Magazine* 29.6 (2012). DOI: 10.1109/MSP.2012.2205597. URL: http://cs224d.stanford.edu/papers/maas_paper.pdf.

[2] J. Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv:1810.04805v2 [cs.CL]* (2018). URL: https://arxiv.org/abs/1810.04805v2.

[3] A. van den Oord et al. “Parallel WaveNet: Fast High-Fidelity Speech Synthesis”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. 2018. URL: https://arxiv.org/abs/1711.10433v1.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2012. DOI: 10.1145/3065386. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[5] J. Ng et al. “Beyond Short Snippets: Deep Networks for Video Classification”. In: *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015. DOI: 10.1109/CVPR.2015.7299101. URL: https://arxiv.org/abs/1503.08909v2.

[6] T. Wang et al. “Video-to-Video Synthesis”. In: *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2018. DOI: 10.5555/3326943.3327049. URL: https://arxiv.org/abs/1808.06601v2.

[7] D. Silver et al. “Mastering the Game of GO with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (2016). DOI: 10.1038/nature16961. URL: http://airesearch.com/wp-content/uploads/2016/01/deepmind-mastering-go.pdf.

[8] M. Moracik et al. “DeepStack: Expert-level Artificial Intelligence in Haeds-up No-limit Poker”. In: *Science* 356.6337 (2017). DOI: 10.1126/science.aam6960. URL: https://arxiv.org/abs/1701.01724.

[9] H. Xue et al. “Deep Matrix Factorization Models for Recommender Systems”. In: *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 2017. DOI: 10.5555/3172077.3172336. URL: https://www.ijcai.org/proceedings/2017/0447.pdf.

[10] F. Fusco et al. “RecoNet: An Interpretable Neural Architecture for Recommender Systems”. In: *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 2019. DOI: 10.24963/ijcai.2019/325. URL: https://www.ijcai.org/proceedings/2019/0325.pdf.

[11] D. Ciresan et al. "Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks". In: *Proc. Int. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2013. DOI: 10.1007/978-3-642-40763-5_51. URL: https://link.springer.com/content/pdf/10.1007%5C%2F978-3-642-40763-5_51.pdf.

[12] A. Estava et al. "Dermatologist-level Classification of Skin Cancer with Deep Neural Networks". In: *Nature* 542.7639 (2017). DOI: 10.1038/nature21056. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8382232/pdf/nihms-1724608.pdf.

[13] N. Ernest et al. "Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions". In: *J. of Defense Management* 6.1 (2016). DOI: : 10.4172/2167-0374.1000144. URL: https://www.longdom.org/open-access/genetic-fuzzy-based-artificial-intelligence-for-unmanned-combat-aerialvehicle-control-in-simulated-air-combat-missions-2167-0374-1000144.pdf.

[14] M. Schwarz et al. "NimbRo Picking: Versatile Part Handling for Warehouse Automation". In: *Proc. Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2017. DOI: 10.1109/ICRA.2017.7989348. URL: http://www.is.uni-bonn.de/papers/ICRA_2017_Schwarz.pdf.

[15] Y. Shrestha, V. Krishna, and G. von Krogh. "Augmenting Organizational Decision-Making with Deep Learning Algorithms: Principles, Promises, and Challenges". In: *J. of Business Research* 123 (2021). DOI: 10.1016/j.jbusres.2020.09.068. URL: https://arxiv.org/abs/2011.02834v1.

[16] D. Lukovnikov et al. "Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level". In: *Proc. Conf. on Word Wide Web (WWW)*. ACM, 2017. DOI: 10.1145/3038912.3052675. URL: https://jens-lehmann.org/files/2017/www_nn_factoid_qa.pdf.

[17] Y. Leviathan and Y. Matias. *Google Duplex: An AI System for Accomplishing Real-World Tasks Over the Phone*. Google AI blog. 2018. URL: https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html.

[18] T. Brown et al. "Language Models are Few-Shot Learners". In: *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2020. URL: https://arxiv.org/abs/2005.14165v4.

[19] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition. Springer, 2009. DOI: 10.1007/978-0-387-84858-7. URL: https://web.stanford.edu/~hastie/Papers/ESLII.pdf.

[20] D. MacKay. *Information Theory, Inference, & Learning Algorithms*. Cambridge University Press, 2003. URL: https://www.inference.org.uk/itprnn/book.pdf.

[21] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. URL: https://link.springer.com/book/9780387310732.

[22] S. Haykin. *Neural Networks and Learning Machines*. 3rd. Pearson, 2009. URL: http://dai.fmph.uniba.sk/courses/NN/haykin.neural-networks.3ed.2009.pdf.

[23] Y. Bengio. “Learning Deep Architectures for AI”. In: *Foundations and Trends in Machine Learning* 2.1 (2009). DOI: 10.1561/2200000006. URL: https://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf.

[24] R. Ramamurthy et al. “Leveraging Domain Knowledge for Reinforcement Learning Using MMC Architectures”. In: *Proc. Int. Conf. on Artificial Neural Networks (ICANN)*. 2019.

[25] M. Lechner et al. “Neural Circuit Policies Enabling Auditable Autonomy”. In: *Nature Machine Intelligence* 2 (2020). DOI: 10.1038/s42256-020-00237-3.

[26] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000. DOI: 10.1007/978-1-4757-3264-1. URL: https://statisticalsupportandresearch.files.wordpress.com/2017/05/vladimir-vapnik-the-nature-of-statistical-learning-springer-2010.pdf.

[27] J. Park. “The Concept of Transition in Quantum Mechanics”. In: *Foundations of Physics* 1.1 (1970). DOI: 10.1007/BF00708652.

[28] C. Bennet. “Logical Reversibility of Computation”. In: *IBM J. of Research and Development* 17.6 (1973). DOI: 10.1147/rd.176.0525.

[29] A. Holevo. “Bounds for the Quantity of Information Transmitted by a Quantum Communication Channel”. In: *Problemy Peredachi Informatsii* 9.3 (1973). URL: http://mi.mathnet.ru/eng/ppi/v9/i3/p3.

[30] R. Ingarden. “Quantum Information Theory”. In: *Reports on Mathematical Physics* 10.1 (1976). DOI: 10.1016/0034-4877(76)90005-7.

[31] P. Benioff. “The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines”. In: *J. of Statistical Physics* 22.5 (1980). DOI: 10.1007/BF01011339.

[32] Y. Manin. *Computable and Noncomputable (in Russian)*. Kibernetika. Moscow: Sovetskoye Radio, 1980.

[33] R. Feynman. “Simulating Physics with Computers”. In: *Int. J. of Theoretical Physics* 10 (1982). DOI: 10.1007/BF02650179.

[34] R. Feynman. “Quantum Mechanical Computers”. In: *Foundations of Physics* 16.6 (1986). DOI: 10.1007/BF01886518.

[35] D. Deutsch and R. Jozsa. “Rapid Solutions of Problems by Quantum Computation”. In: *Proc. of the Royal Society London A* 439 (1992). DOI: 10.1098/rspa.1992.0167.

[36] P. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *Proc. Annual Symp. on Foundations of Computer Science*. IEEE, 1994. DOI: 10.1109/SFCS.1994.365700.

[37] L. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proc. Symp. on Theory of Computing*. ACM, 1996. DOI: 10.1145/237814.237866.

[38] L. Grover. “From Schrödinger’s Equation to the Quantum Search Algorithm”. In: *Quantum Information Processing* 56.2 (2001). DOI: 10.1007/s12043-001-0128-3.

[39] P. Shor. “Polynomial-time Algorithms for Prime Factorization and Discrete Logarithm Problems”. In: *SIAM J. on Computing* 26.5 (1997). DOI: 10.1137/S0036144598347011.

[40] M. Mosca. “Quantum Computer Algorithms”. PhD thesis. University of Oxford, 1999. URL: https://www2.karlin.mff.cuni.cz/~holub/soubory/moscathesis.pdf.

[41] I. Chuang, N. Gershenfeld, and M. Kubinec. “Experimental Implementation of Fast Quantum Searching”. In: *Physical Review Letters* 80.15 (1998). DOI: 10.1103/PhysRevLett.80.3408.

[42] L. Vandersypen et al. “Experimental Realization of Shor’s Quantum Factoring Algorithm Using Nuclear Magnetic Resonance”. In: *Nature* 414.6866 (2001). DOI: 10.1038/414883a.

[43] F. Arute et al. “Quantum Supremacy Using a Programmable Superconducting Processor”. In: *Nature* 574.7779 (2019). DOI: 10.1038/s41586-019-1666-5.

[44] P. Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Academic Press, 2014. DOI: 10.1016/C2013-0-19170-2.

[45] M. Schuld and F. Petruccione. *Machine Learning with Quantum Computers*. Springer, 2021. DOI: 10.1007/978-3-030-83098-4.

[46] V. Dunjiko, J. Taylor, and H. Briegel. “Quantum-Enhanced Machine Learning”. In: *Physical Review Letters* 117.13 (2016). DOI: 10.1103/PhysRevLett.117.130501. URL: https://arxiv.org/abs/1610.08251.

[47] J. Biamonte et al. “Quantum Machine Learning”. In: *Nature* 549.7671 (2017). DOI: 10.1038/nature23474.

[48] J. Abhijith et al. “Quantum Algorithm Implementations for Beginners”. In: *arXiv:1804. 03719 [cs.ET]* (2020). URL: https://arxiv.org/abs/1804.03719v2.

[49] M. Cusumano. “The Business of Quantum Computing”. In: *Communications of the ACM* 61.10 (2018). DOI: 10.1145/3267352. URL: https://cacm.acm.org/magazines/2018/10/231363-the-business-of-quantum-computing/fulltext.

[50] C. Bauckhage et al. *Quantum Machine Learning – An Analysis of Expertise, Research, and Applications*. Fraunhofer Big Data and Artificial Intelligence Alliance. 2020.

[51] M. Bojarski et al. “The NVIDIA PilotNet Experiments”. In: *arXiv:2010.08776 [cs.CV]* (2020). URL: https://arxiv.org/abs/2010.08776v1.

[52] E. Brito et al. "A Hybrid AI Tool to Extract Key Performance Indicators from Financial Reports for Benchmarking". In: *Proc. Symp. on Document Engineering (DocEng)*. ACM, 2019. DOI: 10.1145/3342558.3345420. URL: https://www.researchgate.net/profile/Eduardo-Brito/publication/335944246_A_Hybrid_AI_Tool_to_Extract_Key_Performance_Indicators_from_Financial_Reports_for_Benchmarking/links/5dd299ab4585156b351d2e02/A-Hybrid-AI-Tool-to-Extract-Key-Performance-Indicators-from-Financial-Reports-for-Benchmarking.pdf.

[53] R. Sifa et al. "Towards Automated Auditing with Machine Learning". In: *Proc. Symp. on Document Engineering (DocEng)*. ACM, 2019. DOI: 10.1145/3342558.3345421. URL: https://www.researchgate.net/profile/Birgit-Kirsch/publication/335943303_Towards_Automated_Auditing_with_Machine_Learning/links/5d96fe5d458515c1d391daef/Towards-Automated-Auditing-with-Machine-Learning.pdf.

[54] R. Ramamurthy et al. "ALiBERT: Improved Automated List Inspection (ALi) with BERT". In: *Proc. Symp. on Document Engineering (DocEng)*. ACM, 2021. DOI: 10.1145/3469096.3474928. URL: https://www.researchgate.net/profile/Birgit-Kirsch/publication/335943303_Towards_Automated_Auditing_with_Machine_Learning/links/5d96fe5d458515c1d391daef/Towards-Automated-Auditing-with-Machine-Learning.pdf.

[55] E. Commission. *White Paper on Artificial Intelligence – A European Approach to Excellence and Trust*. COM(2020) 65 final. 2020. URL: https://ec.europa.eu/info/publications/white-paper-artificial-intelligence-european-approach-excellence-and-trust_en.

[56] K. Morik et al. "Yes We Care! – Certification for Machine Learning Methods through the Care Label Framework". In: *arXiv:2105.10197 [cs.LG]* (2021). URL: https://arxiv.org/abs/2105.10197v1.

[57] K. Beckh et al. "Explainable Machine Learning with Prior Knowledge: An Overview". In: *arXiv:2105.10172 [cs.LG]* (2021). URL: https://arxiv.org/abs/2105.10172v1.

[58] A. Holzinger et al. "Digital Transformation for Sustainable Development Goals (SDGs)-A Security, Safety and Privacy Perspective on AI". In: *Proc. Cross-Domain Conf. for Machine Learning and Knowledge Extraction (CD-MAKE)*. 2021. DOI: 10.1007/978-3-030-84060-0_1. URL: http://eprints.cs.univie.ac.at/7013/1/AI-Digital-Transformation-Sustainability-2021.pdf.

[59] E. Wigner. "The Unreasonable Effectiveness of Mathematics in the Natural Sciences". In: *Communications in Pure and Applied Mathematics* 13.1 (1960). DOI: 10.1002/cpa.3160130102. URL: https://www.maths.ed.ac.uk/~v1ranick/papers/wigner.pdf.

[60] F. Wilhelm et al. *Status of Quantum Computer Development*. Whitepaper Federal Office for Information Security. 2020. URL: https://www.bsi.bund.de/EN/Topics/Cryptography/QuantumComputing/quantum_computing.html.

[61] J. Preskill. "Quantum Computing in the NISQ Era and Beyond". In: *Quantum* 2 (2018). DOI: 10.22331/q-2018-08-06-79.

[62] G. Vidal. "Efficient Classical Simulation of Slightly Entangled Quantum Computations". In: *Physical Review Letters* 91.14 (2003). DOI: 10.1103/PhysRevLett.91.147902.

[63] D. Aharanov et al. "Adiabatic Quantum Computation Is Equivalent to Standard Quantum Computation". In: *SIAM Review* 50.4 (2008). DOI: 10.1137/080734479. URL: https://arxiv.org/abs/quant-ph/0405098v2.

[64] A. Mizel, D. Lidar, and M. Mitchell. "Simple Proof of Equivalence between Adiabatic Quantum Computation and the Circuit Model". In: *Physical Review Letters* 99.7 (2007). DOI: 10.1103/PhysRevLett.127.139901. URL: https://arxiv.org/abs/quant-ph/0609067v2.

[65] M. Born and V. Fock. "Beweis des Adiabatensatzes". In: *Zeitschrift für Physik* 51.3–4 (1928). DOI: 10.1007/BF01343193.

[66] T. Albash and D. Lidar. "Adiabatic Quantum Computation". In: *Reviews of Modern Physics* 90.1 (2018). DOI: 10.1103/RevModPhys.90.015002. URL: https://arxiv.org/abs/1611.04471v2.

[67] Z. Bian et al. *The Ising Model: Teaching an Old Problem New Tricks*. Tech. rep. D-Wave Systems, 2010. URL: https://www.dwavesys.com/media/vbklsvbh/weightedmaxsat_v2.pdf.

[68] M. Johnson et al. "Quantum Annealing with Manufactured Spins". In: *Nature* 473.7346 (2011). DOI: 10.1038/nature10012.

[69] T. Lanting et al. "Entanglement in a Quantum Annealing Processor". In: *Physical Review X* 4.021041 (2014). DOI: 10.1103/PhysRevX.4.021041. URL: https://arxiv.org/abs/1401.3500v1.

[70] E. Ising. "Beitrag zur Theorie des Ferromagnetismus". In: *Zeitschrift für Physik* 31.1 (1925). DOI: 10.1007/BF02980577.

[71] J. Hopfield. "Neural Networks and Physical Systems with Collective Computational Abilities". In: *PNAS* 79.8 (1982). DOI: 10.1073/pnas.79.8.2554. URL: https://www.pnas.org/content/pnas/79/8/2554.full.pdf.

[72] A. Lucas. "Ising Formulations of Many NP Problems". In: *Frontiers in Physics* 2 (2014). DOI: 10.3389/fphy.2014.00005. URL: https://arxiv.org/abs/1302.5843v3.

[73] C. Bauckhage et al. "Ising Models for Binary Clustering via Adiabatic Quantum Computing". In: *Proc. Int. Conf. on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*. Springer, 2017. DOI: 10.1007/978-3-319-78199-0_1.

[74] H. Ushijima-Mwesigwa, C. Negre, and S. Mniszewski. "Graph Partitioning Using Quantum Annealing on the D-Wave System". In: *Int. Workshop on Post Moore's Era Supercomputing*. New York: ACM, 2017. DOI: 10.1145/3149526.3149531. URL: https://arxiv.org/abs/1705.03082v1.

[75] C. Bauckhage, R. Sanchez, and R. Sifa. "Problem Solving with Hopfield Networks and Adiabatic Quantum Computing". In: *Proc. Int. Joint Conf. on Neural Networks*. IEEE, 2020. DOI: 10.1109/IJCNN48605.2020.9206916.

[76] E. Farhi et al. "Quantum Computation by Adiabatic Evolution". In: *arXiv:quant-ph/0001106* (2000). URL: https://arxiv.org/abs/quant-ph/0001106v1.

[77] M. Amin. "Effect of Local Minima on Adiabatic Quantum Optimization". In: *Physical Review Letters* 100.13 (2008). DOI: 10.1103/PhysRevLett.100.130503. URL: https://arxiv.org/abs/0709.0528v2.

[78] J. Roland and N. Cerf. "Quantum Search by Local Adiabatic Evolution". In: *Physical Review A* 65.4 (2002). DOI: 10.1103/PhysRevA.65.042308. URL: https://arxiv.org/abs/quant-ph/0107015v1.

[79] C. Bauckhage, R. Sifa, and S. Wrobel. "Adiabatic Quantum Computing for Max-Sum Diversification". In: *Proc. SIAM Int. Conf. on Data Mining (SDM)*. SIAM, 2020. DOI: 10.1137/1.9781611976236.39. URL: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976236.39.

[80] J. Johansson, P. Nation, and F. Nori. "QuTiP 2: A Python Framework for the Dynamics of Open Quantum Systems". In: *Computer Physics Communications* 184.4 (2013). DOI: 10.1016/j.cpc.2012.11.019. URL: https://arxiv.org/abs/1211.6518v1.

[81] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2016. ISBN: 978-1-10-700217-3. URL: https://www.cambridge.org/de/academic/subjects/physics/quantum-physics-quantum-information-and-quantum-computation/quantum-computation-and-quantum-information-10th-anniversary-edition?format=HB.

[82] M. Saeedi and I. Markov. "Synthesis and Optimization of Reversible Circuits – A Survey". In: *ACM Computing Surveys* 45.2 (2013). DOI: 10.1145/2431211.2431220.

[83] M. Freedman et al. "Topological Quantum Computation". In: *Bulletin of the American Mathematical Society* 40.1 (2003). DOI: 10.1090/S0273-0979-02-00964-3.

[84] E. Farhi, J. Goldstone, and S. Gutmann. "A Quantum Approximate Optimization Algorithm". In: *arXiv:1411.4028 [quant-ph]* (2014). URL: https://arxiv.org/abs/1411.4028v1.

[85] A. Peruzzo et al. "A Variational Eigenvalue Solver on a Photonic Quantum Processor". In: *Nature Communications* 5.1 (2014). DOI: 10.1038/ncomms5213.

[86] M. Cerezo et al. "Variational Quantum Algorithms". In: *Nature Review Physics* 3 (2021). DOI: 10.1038/s42254-021-00348-9.

[87] J. Biamonte. “Universal Variational Quantum Computation”. In: *Physical Review A* (2021). DOI: 10.1103/PhysRevA.103.L030401. URL: https://link.aps.org/doi/10.1103/PhysRevA.103.L030401.

[88] W. Wootters and W. Zurek. “A Single Quantum Cannot be Cloned”. In: *Nature* 299.5886 (1982). DOI: 10.1038/299802a0.

[89] F. Leymann. “Towards a Pattern Language for Quantum Algorithms”. In: *Proc. Int. Workshop on Quantum Technology and Optimization Problems (QTOP)*. Springer, 2019. DOI: 10.1007/978-3-030-14082-3_19.

[90] The Qiskit Contributors. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: https:doi.org/10.5281/zenodo.2573505.

[91] The Cirq Contributers. *Cirq, a Python Framework for Creating, Editing, and Invoking Noisy Intermediate Scale Quantum (NISQ) Circuits*. 2021. DOI: 10.5281/zenodo.4750446.

[92] J. Otterbach et al. “Unsupervised Machine Learning on a Hybrid Quantum Computer”. In: *arXiv:1712.05771 [quant-ph]* (2017).

[93] V. Bergholm et al. “PennyLane: Automatic Differentiation of Hybrid Quantum-classical Computations”. In: *arXiv:1811.04968 [quant-ph]* (2018).

[94] R. Sutton and A. Barto. *Reinforcement Learning*. 2nd edition. Bradford Books, 2018.

[95] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 2005.

[96] D. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *arXiv:1312.6114 [stat.ML]* (2014). URL: https://arxiv.org/abs/1312.6114.

[97] I. Goodfellow et al. “Generative Adversarial Nets”. In: *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[98] B. Neyshabur, R. Tomioka, and N. Srebro. “In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning”. In: *arXiv:1412.6614 [cs.LG]* (2014). URL: https://arxiv.org/abs/1412.6614v4.

[99] B. Neal et al. “A Modern Take on the Bias-Variance Tradeoff in Neural Networks”. In: *arXiv:1810.08591 [cs.LG]* (2018). URL: https://arxiv.org/abs/1810.08591v4.

[100] K. van Rijsbergen. *The Geometry of Information Retrieval*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511543333.

[101] G. Birkhoff and J. von Neumann. “The Logic of Quantum Mechanics”. In: *Annals of Mathematics* 37.4 (1936). DOI: 10.2307/1968621.

[102] J. Arrazola et al. “Quantum-inspired Algorithms in Practice”. In: *Quantum* 4 (2020). DOI: 10.22331/q-2020-08-13-307.

[103] A. Frieze, R. Kannan, and S. Vempala. “Fast Monte-Carlo Algorithms for Finding Low-rank Approximations”. In: *J. of the ACM* 51.6 (2004). DOI: 10.1145/1039488.1039494. URL: https://www.math.cmu.edu/~af1p/Texfiles/SVD.pdf.

[104] P. Drineas, R. Kannan, and M. Mahoney. “Fast Monte Carlo Algorithms III: Computing a Compressed Approixmate Matrix Decomposition”. In: *SIAM J. on Computing* (2006). DOI: 10.1137/S0097539704442702. URL: https://www.stat.berkeley.edu/~mmahoney/pubs/matrix3_SICOMP.pdf.

[105] D. Achlioptas and F. McSherry. “Fast Computation of Low-rank Matrix Approximations”. In: *J. of the ACM* 54.9 (2007). DOI: 10.1145/1219092.1219097. URL: https://www.cs.princeton.edu/courses/archive/spr04/cos598B/bib/Mcsherry-svd.pdf.

[106] M. Mahoney and P. Drineas. “CUR Matrix Decompositions for Improved Data Analysis”. In: *PNAS* 106.3 (2009). DOI: 10.1073/pnas.0803205106. URL: https://www.pnas.org/content/pnas/106/3/697.full.pdf.

[107] C. Thurau, K. Kersting, and C. Bauckhage. “Deterministic CUR for Improved Large-Scale Data Analysis: An Empirical Study”. In: *Proc. Int. Conf. on Data Mining*. SIAM, 2012. DOI: 10.1137/1.9781611972825.59. URL: https://epubs.siam.org/doi/pdf/10.1137/1.9781611972825.59.

[108] R. Kannan and S. Vempala. “Randomized Algorithms in Numerical Linear Algebra”. In: *Acta Numerica* 26 (2017). DOI: 10.1017/S0962492917000058. URL: https://www.cc.gatech.edu/~vempala/papers/acta_survey.pdf.

[109] R. Salakhutdinov and A. Mnih. “Probabilistic Matrix Factorization”. In: *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2007. DOI: 10.5555/2981562.2981720. URL: https://papers.nips.cc/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf.

[110] G. Zhang. “Quantum-inspired Evolutionary Algorithms: A Survey and Empirical Study”. In: *J. of Heuristics* 17 (2011). DOI: 10.1007/s10732-010-9136-0.

[111] E. Tang. “A Quantum-Inspired Classical Algorithm for Recommendation Systems”. In: *Proc. Symp. on the Theory of Computing (STOC)*. ACM, 2019. DOI: 10.1145/3313276.3316310.

[112] I. Kerenidis and A. Prakash. “Quantum Recommendation Systems”. In: *Proc. Innovations in Theoretical Computer Science Conf. (ITCS)*. 2017. DOI: 10.4230/LIPIcs.ITCS.2017.49. URL: https://arxiv.org/abs/1603.08675v3.

[113] A. Kitaev. “Quantum Measurements and the Abelian Stabilizer Problem”. In: *arXiv:quant-ph/9511026* (1995). URL: https://arxiv.org/abs/quant-ph/9511026.

[114] R. Cleve et al. “Quantum Algorithms Revisited”. In: *Proc. of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998). DOI: 10.1098/rspa.1998.0164. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1998.0164.

[115] D. S. Abrams and S. Lloyd. “Quantum Algorithm Providing Exponential Speed Increase for Finding Eigenvalues and Eigenvectors”. In: *Physical Review Letters* 83.24 (1999). DOI: `10.1103/PhysRevLett.83.5162`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.83.5162`.

[116] N. Chia et al. “Quantum-inspired Algorithms for Solving Low-rank Linear Equation Systems with Logarithmic Dependence on the Dimension”. In: *Int. Symp. on Algorithms and Computation (ISAAC)*. 2020. DOI: `10.4230/LIPIcs.ISAAC.2020.47`.

[117] A. Gilyen et al. “Quantum Singular Value Transformation and Beyond: Exponential Improvements for Quantum Matrix Arithmetics”. In: *Proc. Symp. on the Theory of Computing (STOC)*. ACM, 2019. DOI: `10.1145/3313276.3316366`.

[118] N. Chia, H. Lin, and C. Wang. “Quantum-inspired Sublinear Classical Algorithms for Solving Low-rank Linear Systems”. In: *arXiv:1811.04852 [cs.DS]* (2018).

[119] A. W. Harrow, A. Hassidim, and S. Lloyd. “Quantum Algorithm for Linear Systems of Equations”. In: *Physical Review Letters* 103 (15 2009). DOI: `10.1103/PhysRevLett.103.150502`. URL: `https://arxiv.org/abs/0811.3171v3`.

[120] N. Chepurko et al. “Quantum-Inspired Algorithms from Randomized Numerical Linear Algebra”. In: *arXiv:2011.04125 [cs.DS]* (2020).

[121] R. Orus. “A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States”. In: *Annals of Physics* 349 (2014). DOI: `10.1016/j.aop.2014.06.013`. URL: `https://arxiv.org/abs/1306.2164`.

[122] J. Biamonte and V. Bergholm. “Tensor Networks in a Nutshell”. In: *arXiv:1708.00006 [quant-ph]* (2017).

[123] A. Cichocki et al. “Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions”. In: *Foundations and Trends in Machine Learning* 9.4–5 (2016). DOI: `10.1561/2200000059`.

[124] A. Cichocki et al. “Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 2 Applications and Future Perspectives”. In: *Foundations and Trends in Machine Learning* 9.6 (2017). DOI: `10.1561/2200000067`.

[125] L. De Lathauwer, B. De Moor, and J. Vanderwalle. “A Multilinear Singular Value Decomposition”. In: *SIAM J. on Matrix Analysis and Applications* 21.4 (2000). DOI: `/10.1137/S0895479896305696`.

[126] R. Sifa et al. “Matrix and Tensor Factorization Based Game Content Recommender Systems: A Bottom-Up Architecture and a Comparative Online Evaluation”. In: *Proc. Conf. on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI, 2018.

[127] C. Bauckhage et al. “Fast Learning for Customizable Head Pose Recognition in Robotic Wheelchair Control”. In: *Proc. Int. Conf. Automatic Face and Gesture Recognition (FG)*. IEEE, 2006.

[128] L. Hillebrand et al. "Interpretable Topic Extraction and Word Embedding Learning Using Non-Negative Tensor DEDICOM". In: *Machine Learning and Knowledge Extraction* 3.1 (2021). DOI: `10.3390/make3010007`.

[129] H. Lin, M. Tegmark, and D. Rolnick. "Why Does Deep and Cheap Learning Work So Well?" In: *J. of Statistical Physics* 168 (2017). DOI: `10.1007/s10955-017-1836-5`.

[130] E. Stoudenmire and D. Schwab. "Supervised Learning with Tensor Networks". In: *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[131] I. Glasser, N. Pancotti, and J. Cirac. "From Probabilistic Graphical Models to Generalized Tensor Networks for Supervised Learning". In: *IEEE Access* 8 (2020). DOI: `10.1109/ACCESS.2020.2986279`.

[132] C. Roberts et al. "TensorNetwork: A Library for Physics and Machine Learning". In: *arXiv:1905.01330 [physics.comp-ph]* (2019).

[133] S. Mücke, N. Piatkowski, and K. Morik. "Hardware Acceleration of Machine Learning Beyond Linear Algebra". In: *Proc. European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*. 2019. DOI: `10.1007/978-3-030-43823-4_29`.

[134] J. Boyd. "Fujitsu's CMOS Digital Annealer Produces Quantum Computer Speeds". In: *IEEE Spectrum* May (2018).

[135] M. Aramon et al. "Physics-Inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer". In: *Frontiers in Physics* 7 (2019). DOI: `10.3389/fphy.2019.00048`. URL: `https://www.frontiersin.org/article/10.3389/fphy.2019.00048`.

[136] S. Mücke, N. Piatkowski, and K. Morik. "Learning Bit by Bit: Extracting the Essence of Machine Learning". In: *Proc. Conf. Learning, Knowledge, Data, Analytics (KDML-LWDA)*. 2019. URL: `http://ceur-ws.org/Vol-2454/paper_51.pdf`.

[137] D. Horn and A. Gottlieb. "Algorithm for Data Clustering in Pattern Recognition Problems Based on Quantum Mechanics". In: *Physical Review Letters* 88.1 (2001). DOI: `10.1103/PhysRevLett.88.018702`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.88.018702`.

[138] M. Weinstein and D. Horn. "Dynamic Quantum Clustering: A Method for Visual Exploration of Structures in Data". In: *Physical Review E* 80.6 (2009). DOI: `10.1103/PhysRevE.80.066117`. URL: `https://link.aps.org/doi/10.1103/PhysRevE.80.066117`.

[139] H. Nezamabadi-pour. "A Quantum-inspired Gravitational Search Algorithm for Binary Encoded Optimization Problems". In: *Engineering Applications of Artificial Intelligence* 4 (2015). DOI: `10.1016/j.engappai.2015.01.002`.

[140] J. Lou et al. "Failure Prediction by Relevance Vector Regression with Improved Quantum-inspired Gravitational Search". In: *J. of Network and Computer Applications* 103 (2018). DOI: `10.1016/j.jnca.2017.11.013`.

[141] B. Rubinstein. “Evolving Quantum Circuits Using Genetic Programming”. In: *Proc. Congress on Evolutionary Computation (CEC)*. IEEE, 2001. DOI: 10.1109/CEC.2001.934383.

[142] A. Leier. “Evolution of Quantum Algorithms using Genetic Programming”. PhD thesis. University of Dortmund, 2004.

[143] L. Franken et al. “Gradient-free Quantum Optimization on NISQ Devices”. In: *arXiv:2012.13453 [quant-ph]* (2020).

[144] D. Panchenko. *The Sherrington-Kirkpatrick Model*. Springer, 2013. DOI: 10.1007/978-1-4614-6289-7.

[145] K. McKiernan et al. “Automated Quantum Programming via Reinforcement Learning for Combinatorial Optimization”. In: *arXiv:1908.08054 [quant-ph]* (2019).

[146] K. Guy and G. Perdue. *Using Reinforcement Learning to Optimize Quantum Circuits in the Presence of Noise*. Tech. rep. Batavia, IL, USA: Fermi National Accelerator Lab, 2020.

[147] T. Fösel et al. “Quantum Circuit Optimization with Deep Reinforcement Learning”. In: *arXiv:2103.07585 [quant-ph]* (2021).

[148] L. Cincio et al. “Machine Learning of Noise-Resilient Quantum Circuits”. In: *Physical Review X Quantum* 2 (1 2021). DOI: 10.1103/PRXQuantum.2.010324. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.010324.

[149] M. Pirhooshyaran and T. Terlaky. “Quantum Circuit Design Search”. In: *Quantum Machine Intelligence* 3.25 (2021). DOI: 10.1007/s42484-021-00051-z.

[150] A. Zulehner, A. Paler, and R. Wille. “An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures”. In: *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 38.7 (2019). DOI: 10.1109/TCAD.2018.2846658.

[151] G. Li, Y. Ding, and Y. Xie. “Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices”. In: *Proc. Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, 2019. DOI: 10.1145/3297858.3304023. URL: https://doi.org/10.1145/3297858.3304023.

[152] G. Acampora and R. Schiattarella. “Deep Neural Networks for Quantum Circuit Mapping”. In: *Neural Computing and Applications* 33 (2021). DOI: 10.1007/s00521-021-06009-3.

[153] J. Kusyk, S. Saeed, and M. Uyar. “Survey on Quantum Circuit Compilation for Noisy Intermediate-Scale Quantum Computers: Artificial Intelligence to Heuristics”. In: *IEEE Trans. on Quantum Engineering* 2 (2021). DOI: 10.1109/TQE.2021.3068355.

[154] L. K. Grover and T. Rudolph. “Creating superpositions that correspond to efficiently integrable probability distributions”. In: *CoRR* abs/quant-ph/0208112v1 (2002). arXiv: 0208112v1. URL: https://arxiv.org/abs/quant-ph/0208112v1.

[155] C. Zoufal, A. Lucchi, and S. Woerner. "Quantum Generative Adversarial Networks for learning and loading random distributions". In: *npj Quantum Information* 5.1 (2019). DOI: `10.1038/s41534-019-0223-2`. URL: `https://arxiv.org/abs/1904.00043v2`.

[156] R. Harper, S. T. Flammia, and J. J. Wallman. "Efficient Learning of Quantum Noise". In: *Nature Physics* 16.12 (2020). DOI: `10.1038/s41567-020-0992-8`. URL: `https://arxiv.org/abs/1907.13022v2`.

[157] J. M. Martinis. "Qubit metrology for building a fault-tolerant quantum computer". In: *npj Quantum Information* 1.1 (2015). ISSN: 2056-6387. DOI: `10.1038/npjqi.2015.5`.

[158] E. Aïmeur, G. Brassard, and S. Gambs. "Machine Learning in a Quantum World". In: *Proc. Canadian Conf. on Artificial Intelligence*. 2006. DOI: `10.1007/11766247_37`.

[159] D. Riste et al. "Demonstration of Quantum Advantage in Machine Learning". In: *npj Quantum Information* 3.16 (2017). DOI: `10.1038/s41534-017-0017-3`.

[160] W. O'Quinn and S. Mao. "Quantum Machine Learning: Recent Advances and Outlook". In: *IEEE Wireless Communications* 27.3 (2020). DOI: `10.1109/MWC.001.1900341`.

[161] D. Coppersmith. "An Approximate Fourier Transform Useful in Quantum Factoring". In: *arXiv:quant-ph/0201067* (2002). URL: `https://arxiv.org/abs/quant-ph/0201067`.

[162] A. Ambainis. "Quantum Walk Algorithm for Element Distinctness". In: *Symp. on Foundations of Computer Science(FOCS)*. IEEE, 2004. DOI: `10.1109/FOCS.2004.54`.

[163] M. Szegedy. "Quantum Speed-up of Markov Chain Based Algorithms". In: *Symp. on Foundations of Computer Science (FOCS)*. IEEE, 2004. DOI: `10.1109/FOCS.2004.53`.

[164] L. Grover and T. Rudolph. "Creating Superpositions that Correspond to Efficiently Integrable Probability Distributions". In: *arXiv:quant-ph/0208112* (2002). URL: `https://arxiv.org/abs/quant-ph/0208112`.

[165] A. M. Childs, R. Kothari, and R. D. Somma. "Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision". In: *SIAM Journal on Computing* 46.6 (2017). DOI: `10.1137/16M1087072`.

[166] X.-D. Cai et al. "Experimental Quantum Computing to Solve Systems of Linear Equations". In: *Physical Review Letters* 110.23 (2013). DOI: `10.1103/PhysRevLett.110.230501`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.110.230501`.

[167] J. Pan et al. "Experimental Realization of Quantum Algorithm for Solving Linear Systems of Equations". In: *Physical Review A* 89.2 (2014). DOI: `10.1103/PhysRevA.89.022313`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.89.022313`.

[168] Z. Zhao et al. "Bayesian Deep Learning on a Quantum Computer". In: *Quantum Machine Intelligence* 1 (2019). DOI: `10.1007/s42484-019-00004-7`.

[169] G. Golub and J. van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[170] P. Rebentrost et al. "Quantum Singular-Value Decomposition of Nonsparse Low-rank Matrices". In: *Phys. Rev. A* 97.1 (2018). DOI: 10.1103/PhysRevA.97.012327. URL: https://link.aps.org/doi/10.1103/PhysRevA.97.012327.

[171] G. Low and I. Chuang. "Hamiltonian Simulation by Qubitization". In: *Quantum* 3 (2019). DOI: 10.22331/q-2019-07-12-163.

[172] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. "Preconditioned Quantum Linear System Algorithm". In: *Physical Review Letters* 110 (2013). DOI: 10.1103/PhysRevLett.110.250504. URL: https://arxiv.org/abs/1301.2340v4.

[173] H.-Y. Huang, K. Bharti, and P. Rebentrost. "Near-term quantum algorithms for linear systems of equations". In: *arXiv.org [quant-ph]* (2019). URL: https://arxiv.org/abs/1909.07344v2.

[174] Y. Subas, R. D. Somma, and D. Orsucci. "Quantum Algorithms for Systems of Linear Equations Inspired by Adiabatic Quantum Computing". In: *Physical Review Letters* 122 (6 2019). DOI: 10.1103/PhysRevLett.122.060504. URL: https://arxiv.org/abs/1805.10549v2.

[175] Y. Lee, J. Joo, and S. Lee. "Hybrid quantum linear equation algorithm and its experimental test on IBM Quantum Experience". In: *Scientific Reports* 9 (2019). ISSN: 2045-2322. DOI: 10.1038/s41598-019-41324-9. URL: https://www.nature.com/articles/s41598-019-41324-9.pdf.

[176] C. Bravo-Prieto et al. "Variational Quantum Linear Solver". In: *arXiv.org [quant-ph]* (2020). URL: https://arxiv.org/abs/1909.05820v2.

[177] X. Xu et al. "Variational algorithms for linear algebra". In: *Science Bulletin* 66.21 (2021). ISSN: 2095-9273. DOI: 10.1016/j.scib.2021.06.023. URL: https://arxiv.org/abs/1909.03898v2.

[178] N. Wiebe, D. Braun, and S. Lloyd. "Quantum Algorithm for Data Fitting". In: *Physical Review Letters* 109.5 (2012). DOI: 10.1103/PhysRevLett.109.050505. URL: https://link.aps.org/doi/10.1103/PhysRevLett.109.050505.

[179] M. Schuld, I. Sinayskiy, and F. Petruccione. "Prediction by Linear Regression on a Quantum Computer". In: *Physical Review A* 94 (2016). DOI: 10.1103/PhysRevA.94.022342. URL: https://arxiv.org/abs/1601.07823v2.

[180] G. Wang. "Quantum Algorithm for Linear Regression". In: *Physical Review A* 96 (2017). DOI: 10.1103/PhysRevA.96.012335. URL: https://arxiv.org/abs/1402.0660v6.

[181] C.-H. Yu, F. Gao, and Q.-Y. Wen. "An Improved Quantum Algorithm for Ridge Regression". In: *IEEE Trans. on Knowledge and Data Engineering* 33.3 (2021). DOI: 10.1088/1674-1056/ac1b84.

[182] Y.-Y. Hou et al. “Quantum Partial Least Squares Regression Algorithm for Multiple Correlation Problem”. In: *Chinese Physics B* (2021). DOI: 10.1088/1674-1056/ac1b84.

[183] Y. Liu and S. Zhang. “Fast Quantum Algorithms for Least Squares Regression and Statistic Leverage Scores”. In: *Theoretical Computer Science* 657 (2016). DOI: 10.1016/j.tcs.2016.05.044. URL: http://www.cse.cuhk.edu.hk/~syzhang/papers/QLS.pdf.

[184] A. Gilyen, Z. Song, and E. Tang. “An Improved Quantum-inspired Algorithm for Linear Regression”. In: *arXiv:2009.07268 [cs.DS]* (2020). URL: https://arxiv.org/abs/2009.07268.

[185] P. Date and T. Potok. “Adiabatic Quantum Linear Regression”. In: *Scientific Reports* 11 (2021). DOI: 110.1038/s41598-021-01445-6. URL: https://www.nature.com/articles/s41598-021-01445-6.pdf.

[186] D. Aloise et al. “NP-Hardness of Euclidean Sum-of-Squares Clustering”. In: *Machine Learning* 75.2 (2009). DOI: 10.1007/s10994-009-5103-0. URL: https://link.springer.com/content/pdf/10.1007/s10994-009-5103-0.pdf.

[187] C. Bauckhage et al. “Adiabatic Quantum Computing for Kernel k=2 Means Clustering”. In: *Proc. Conf. Learning, Knowledge, Data, Analytics (KDML-LWDA)*. 2018. DOI: http://publica.fraunhofer.de/documents/N-520827.html. URL: http://ceur-ws.org/Vol-2191/paper3.pdf.

[188] M. Jünger et al. “Performance of a Quantum Annealer for Ising Ground State Computations on Chimera Graphs”. In: *arXiv:1904.11965 [cs.DS]* (2019). URL: https://arxiv.org/abs/1904.11965v1.

[189] D. Arthur and P. Date. “Balanced k-Means Clustering on an Adiabatic Quantum Computer”. In: *Quantum Information Processing* 20 (2021). DOI: 10.1007/s11128-021-03240-8. URL: https://arxiv.org/abs/2008.04419v1.

[190] P. Date, D. Arthur, and L. Pusey-Nazzaro. “QUBO Formulations for Training Machine Learning Models”. In: *Scientific Reports* 11 (2021). DOI: 10.1038/s41598-021-89461-4. URL: https://www.nature.com/articles/s41598-021-89461-4.pdf.

[191] C. Bauckhage et al. “A QUBO Formulation of the k-Medoids Problem”. In: *Proc. Conf. Learning, Knowledge, Data, Analytics (KDML-LWDA)*. 2019. DOI: http://publica.fraunhofer.de/dokumente/N-562211.html. URL: http://ceur-ws.org/Vol-2454/paper_39.pdf.

[192] S. W. Hong et al. “Market Graph Clustering via QUBO and Digital Annealing”. In: *J. of Risk and Financial Management* 14.1 (2021). DOI: 10.3390/jrfm14010034. URL: https://www.mdpi.com/1911-8074/14/1/34.

[193] E. Aïmeur, G. Brassard, and S. Gambs. “Quantum Clustering Algorithms”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. 2007. DOI: 10.1145/1273496.1273497. URL: https://icml.cc/imls/conferences/2007/proceedings/papers/518.pdf.

[194] E. Aïmeur, G. Brassard, and S. Gambs. “Quantum Speed-up for Unsupervised Learning”. In: *Machine Learning* 90.2 (2013). DOI: 10.1007/s10994-012-5316-5.

[195] T. Tomesh et al. “Coreset Clustering on Small Quantum Computers”. In: *Electronics* (2021). DOI: 10.3390/electronics10141690. URL: https://www.mdpi.com/2079-9292/10/14/1690.

[196] N. Wiebe, A. Kapoor, and K. Svore. “Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning”. In: *Quantum Information & Computation* 15.3–4 (2015). DOI: 10.5555/2871393.2871400. URL: https://arxiv.org/abs/1401.2142v2.

[197] C. Dürr and P. Høyer. “A Quantum Algorithm for Finding the Minimum”. In: *arXiv:quant-ph/9607014* (1999).

[198] S. Lloyd, M. Mohseni, and P. Rebentrost. “Quantum Algorithms for Supervised and Unsupervised Machine Learning”. In: *arXiv:1307.0411 [quant-ph]* (2013). URL: https://arxiv.org/abs/1307.0411v2.

[199] A. Basheer, A. Afham, and S. Goyal. “Quantum k Nearest Neighbor Algorithm”. In: *arXiv:2003.09187 [quant-ph]* (2020). URL: https://arxiv.org/abs/2003.09187.

[200] K. Mitarai, M. Kitagawa, and K. Fujii. “Quantum Analog-Digital Conversion”. In: *Physical Review A* 99.1 (2019). DOI: 10.1103/PhysRevA.99.012301. URL: https://link.aps.org/doi/10.1103/PhysRevA.99.012301.

[201] M. Schuld, M. Fingerhuth, and F. Petruccione. “Implementing a Distance-based Classifier with a Quantum Interference Circuit”. In: *Europhysics Letters* 119.6 (2017). DOI: 10.1209/0295-5075/119/60002.

[202] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002. URL: https://ieeexplore.ieee.org/servlet/opac?bknumber=6267332.

[203] M. Schuld, I. Sinayskiy, and F. Petruccione. “Quantum Computing for Pattern Classification”. In: *Proc. Pacific Rim Int. Conf. on Artificial Intelligence (PRICAI).* Springer, 2014. DOI: 10.1007/978-3-319-13560-1_17.

[204] M. Schuld, I. Sinayskiy, and F. Petruccione. “Simulating a Perceptron on a Quantum Computer”. In: *Physics Letters A* 379.7 (2015). DOI: https://doi.org/10.1016/j.physleta.2014.11.061. URL: https://www.sciencedirect.com/science/article/pii/S037596011401278X.

[205] C. A. Trugenberger. “Probabilistic Quantum Memories”. In: *Physical Review Letters* (2001). DOI: 10.1103/PhysRevLett.87.067901. URL: https://link.aps.org/doi/10.1103/PhysRevLett.87.067901.

[206] C. A. Trugenberger. “Quantum Pattern Recognition”. In: *Quantum Information Processing* 1.6 (2002). DOI: 10.1023/A:1024022632303.

[207] N. Wiebe, A. Kapoor, and K. Svore. “Quantum Perceptron Models”. In: *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2016. DOI: 10.5555/3157382.3157545. URL: https://arxiv.org/abs/1602.04799v1.

[208] F. Tacchino et al. “An Artificial Neuron Implemented on an Actual Quantum Processor”. In: *npj Quantum Information* 5 (2019). DOI: 10.1038/s41534-019-0140-4.

[209] M. Schuld and N. Killoran. “Quantum Machine Learning in Feature Hilbert Spaces”. In: *Physical Review Letters* 122.4 (2019). DOI: 10.1103/PhysRevLett.122.040504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.122.040504.

[210] E. Grant et al. “Hierarchical Quantum Classifiers”. In: *njp Quantum Information* 4 (2018). DOI: 10.1038/s41534-018-0116-9.

[211] W. Huggins et al. “Towards Quantum Machine Learning with Tensor Networks”. In: *Quantum Science and Technology* 4.2 (2019). DOI: 10.1088/2058-9565/aaea94.

[212] Y. Freund and R. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: *J. of Computer and System Sciences* 55.1 (1997). DOI: 10.1006/jcss.1997.1504. URL: https://www.cis.upenn.edu/~mkearns/teaching/COLT/adaboost.pdf.

[213] P. Viola and M. Jones. “Robust Real-Time Face Detection”. In: *Int. J. of Computer Vision* 57 (2004). DOI: 10.1023/B:VISI.0000013087.49260.fb. URL: https://www.researchgate.net/profile/Michael-Jones-66/publication/220660094_Robust_Real-Time_Face_Detection/links/02bfe50d33d12e86ed000000/Robust-Real-Time-Face-Detection.pdf.

[214] H. Neven et al. “QBoost: Large Scale Classifier Training with Adiabatic Quantum Optimization”. In: *Proc. Asian Conf. on Machine Learning (ACML)*. 2012. URL: http://proceedings.mlr.press/v25/neven12/neven12.pdf.

[215] K. Pudenz and D. Lidar. “Quantum Adiabatic Machine Learning”. In: *Quantum Information Processing* 12.5 (2013). DOI: 10.1007/s11128-012-0506-4.

[216] M. Schuld and F. Petruccione. “Quantum Ensembles of Quantum Classifiers”. In: *Scientific Reports* 8.2772 (2018). DOI: 10.1038/s41598-018-20403-3.

[217] B. E. Boser, I. Guyon, and V. Vapnik. “A Training Algorithm for Optimal Margin Classifiers”. In: *Proc. Conf. on Computational Learning Theory (COLT)*. 1992. DOI: 10.1145/130385.130401. URL: https://www.researchgate.net/profile/Bernhard-Boser/publication/2376111_A_Training_Algorithm_for_Optimal_Margin_Classifier/links/560eccc208ae0fc513ee8fc9/A-Training-Algorithm-for-Optimal-Margin-Classifier.pdf.

[218] V. Havlicek et al. “Supervised Learning with Quantum-enhanced Feature Spaces”. In: *Nature* 567.7747 (2019). DOI: 10.1038/s41586-019-0980-2.

[219] P. Wittek. “Supervised Learning and Support Vector Machines”. In: *Quantum Machine Learning*. Academic Press, 2014. DOI: `10.1016/B978-0-12-800953-6.00015-3`. URL: `https://www.sciencedirect.com/science/article/pii/B9780128009536000153`.

[220] H. K. Wang et al. “Application of the Least Squares Support Vector Machine Based on Quantum Particle Swarm Optimization for Data Fitting of Small Samples”. In: *Mechanical Science and Engineering IV*. Vol. 472. 2014. DOI: `10.4028/www.scientific.net/AMM.472.485`.

[221] P. Rebentrost, M. Mohseni, and S. Lloyd. “Quantum Support Vector Machine for Big Data Classification”. In: *Physical Review Letters* 113 (2014). DOI: `10.1103/physrevlett.113.130503`.

[222] C. Havenstein, D. Thomas, and S. Chandrasekaran. “Comparisons of Performance between Quantum and Classical Machine Learning”. In: *SMU Data Science Review* 1.4 (2018).

[223] P. V. Zahorodko et al. “Comparisons of performance between quantum-enhanced and classical machine learning algorithms on the IBM Quantum Experience”. In: *Journal of Physics: Conference Series* 1840 (2021). DOI: `10.1088/1742-6596/1840/1/012021`.

[224] K. Hornik, M. Stinchcombe, and H. White. “Multilayer Feedforward Networks are Universal Approximators”. In: *Neural Networks* 2 (1989).

[225] D. Rumelhart, G. Hinton, and R. Williams. “Learning Representations by Backpropagating Errors”. In: *Nature* 323.6088 (1986). DOI: `10.1038/323533a0`.

[226] J. Allcock et al. “Quantum Algorithms for Feedforward Neural Networks”. In: *ACM Trans. on Quantum Computing* (2018). DOI: `10.1145/3411466`. URL: `https://arxiv.org/abs/1812.03089v2`.

[227] K. Beer et al. “Training deep quantum neural networks”. In: *Nature Communications* 11 (2020). DOI: `10.1038/s41467-020-14454-2`. URL: `https://www.nature.com/articles/s41467-020-14454-2.pdf`.

[228] K. Mitarai et al. “Quantum Circuit Learning”. In: *Physical Review A* 98.3 (2018). DOI: `10.1103/PhysRevA.98.032309`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.98.032309`.

[229] G. Verdon et al. “Learning to learn with quantum neural networks via classical neural networks”. In: *arXiv:1907.05415 [quant-ph]* (2019). URL: `https://arxiv.org/abs/1907.05415v1`.

[230] J. R. McClean et al. “Barren Plateaus in Quantum Neural Network Training Landscapes”. In: *Nature Communications* 9 (2018). DOI: `10.1038/s41467-018-07090-4`.

[231] D. Zhu et al. “Training of Quantum Circuits on a Hybrid Quantum Computer”. In: *Science Advances* 5.10 (2019). DOI: `10.1126/sciadv.aaw9918`.

[232] S. Debnath et al. "Demonstration of a Small Programmable Quantum Computer with Atomic Qubits". In: *Nature* 4.536 (2016). DOI: 10.1038/nature18648.

[233] V. Leyton-Ortega, A. Perdomo-Ortiz, and O. Perdomo. "Robust Implementation of Generative Modeling with Parametrized Quantum Circuits". In: *Quantum Machine Intelligence* 3 (2020). DOI: 10.1007/s42484-021-00040-2.

[234] M. Rossi et al. "Quantum Hypergraph States". In: *New J. of Physics* 15.11 (2013). DOI: 10.1088/1367-2630/15/11/113022. URL: https://doi.org/10.1088/1367-2630/15/11/113022.

[235] Z. Zhao, J. K. Fitzsimons, and J. F. Fitzsimons. "Quantum-assisted Gaussian Process Regression". In: *Phys. Rev. A* 99.5 (2019). DOI: 10.1103/PhysRevA.99.052331. URL: https://link.aps.org/doi/10.1103/PhysRevA.99.052331.

[236] P. Dayan et al. "The Helmholtz Machine". In: *Neural Computation* 7.5 (1995). DOI: 10.1162/neco.1995.7.5.889. URL: https://doi.org/10.1162/neco.1995.7.5.889.

[237] G. Hinton et al. "The "Wake-Sleep" Algorithm for Unsupervised Neural Networks". In: *Science* 268.5214 (1995). DOI: 10.1126/science.7761831. URL: https://www.science.org/doi/abs/10.1126/science.7761831.

[238] T. van Dam et al. "Hybrid Helmholtz Machines: A Gate-based Quantum Circuit Implementation". In: *Quantum Information Processing* 19 (2020). DOI: 10.1007/s11128-020-02660-2.

[239] M. H. Amin et al. "Quantum Boltzmann Machine". In: *Phys. Rev. X* 8.2 (2018). DOI: 10.1103/PhysRevX.8.021050. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.021050.

[240] V. Dixit et al. "Training Restricted Boltzmann Machines With a D-Wave Quantum Annealer". In: *Frontiers in Physics* 9 (2021). DOI: 10.3389/fphy.2021.589626. URL: https://www.frontiersin.org/article/10.3389/fphy.2021.589626.

[241] S. Cheng, J. Chen, and L. Wang. "Information Perspective to Probabilistic Modeling: Boltzmann Machines versus Born Machines". In: *Entropy* 20 (2018). DOI: 10.3390/e20080583.

[242] J.-G. Liu and L. Wang. "Differentiable Learning of Quantum Circuit Born Machines". In: *Physical Review A* 98.6 (2018). DOI: 10.1103/PhysRevA.98.062324. URL: https://arxiv.org/abs/1804.04168.

[243] B. Coyle et al. "The Born supremacy: quantum advantage and training of an Ising Born machine". In: *njp Quantum Information* 6 (2019). DOI: 10.1038/s41534-020-00288-9.

[244] S. Y.-C. Chen and S. Yoo. "Federated Quantum Machine Learning". In: *Entropy* (2021). DOI: 10.3390/e23040460. URL: https://arxiv.org/abs/2103.12010v1.

[245] Y.-B. Sheng and L. Zhou. "Distributed secure quantum machine learning". In: *Science Bulletin* 62 (2017). DOI: 10.1016/j.scib.2017.06.007.

[246] M. Benedetti et al. “Parameterized Quantum Circuits as Machine Learning Models”. In: *Quantum Science and Technology* 4.4 (2019). DOI: `10.1088/2058-9565/ab4eb5`.

[247] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik. “Simulation of Electronic Structure Hamiltonians Using Quantum Computers”. In: *Molecular Physics* 109.5 (2011). DOI: `10.1080/00268976.2011.552441`.

[248] A. Dalzell et al. “How Many Qubits Are Needed for Quantum Computational Supremacy?” In: *Quantum* 4 (2020). DOI: `10.22331/q-2020-05-11-264`.

[249] V. Akshay et al. “Reachability Deficits in Quantum Approximate Optimization”. In: *Physical Review Letters* 124.9 (2020). DOI: `10.1103/PhysRevLett.124.090504`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.124.090504`.

[250] J. Cirac. “Quantum Computing and Simulation: Where We Stand and What Awaits Us”. In: *Nanophotonics* 10.1 (2021). DOI: `doi:10.1515/nanoph-2020-0351`. URL: `https://doi.org/10.1515/nanoph-2020-0351`.

[251] H. Chen et al. “Universal Discriminative Quantum Neural Networks”. In: *Quantum Machine Intelligence* 3.1 (2020). DOI: `10.1007/s42484-020-00025-7`.

[252] S. Aaronson. “The Learnability of Quantum States”. In: *Proc. of the Royal Society A: Mathematical, Physical and Engineering Sciences* 463.2088 (2007). DOI: `10.1098/rspa.2007.0113`. URL: `https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2007.0113`.

[253] A. Rocchetto et al. “Experimental Learning of Quantum States”. In: *Science Advances* (2019). DOI: `10.1126/sciadv.aau1946`. URL: `https://www.science.org/doi/abs/10.1126/sciadv.aau1946`.

[254] Ö. Legeza and J. Sólyom. “Quantum data compression, quantum information generation, and the density-matrix renormalization-group method”. In: *Physical Review B* 70 (2004). DOI: `10.1103/PhysRevB.70.205118`.

[255] J. Romero, J. P. Olson, and A. Aspuru-Guzik. “Quantum Autoencoders for Efficient Compression of Quantum Data”. In: *Quantum Science and Technology* 2.4 (2017). DOI: `10.1088/2058-9565/aa8072`. URL: `https://doi.org/10.1088/2058-9565/aa8072`.

[256] Y. Ding et al. “Experimental Implementation of a Quantum Autoencoder via Quantum Adders”. In: *Advanced Quantum Technologies* 2.7–8 (2019). DOI: `https://doi.org/10.1002/qute.201800065`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201800065`.

[257] M. Benedetti et al. “Adversarial Quantum Circuit Learning for Pure State Approximation”. In: *New Journal of Physics* 21.4 (2019). DOI: `10.1088/1367-2630/ab14b5`. URL: `https://doi.org/10.1088/1367-2630/ab14b5`.

[258] L. Hu et al. “Quantum Generative Adversarial Learning in a Superconducting Quantum Circuit”. In: *Science Advances* 5.1 (2019). DOI: `10.1126/sciadv.aav2761`. URL: `https://www.science.org/doi/abs/10.1126/sciadv.aav2761`.

[259] B. Schumacher and M. A. Nielsen. “Quantum Data Processing and Error Correction”. In: *Physical Review A* 54.4 (1996).

[260] S. Aaronson. “Read the Fine Print”. In: *Nature Physics* 11.4 (2015). DOI: `10.1038/nphys3272`.

[261] A. Childs. “Equation Solving by Simulation”. In: *Nature Physics* 5.12 (2009). DOI: `10.1038/nphys1473`.

[262] O. Egecioglu, H. Ferhatosmanoglu, and U. Ogras. “Dimensionality Reduction and Similarity Computation by Inner-product Approximations”. In: *IEEE Trans. on Knowledge and Data Engineering* 16.6 (2004). DOI: `10.1109/TKDE.2004.9`.

[263] *Tuning CUDA Applications for Ampere*. Application Note. 2788 San Tomas Expressway, Santa Clara, CA 95051: NVIDIA Corporation, 2021. URL: `https://docs.nvidia.com/cuda/pdf/Ampere_Tuning_Guide.pdf`.

[264] M. Frigo et al. “Cache-Oblivious Algorithms”. In: *Foundations of Computer Science*. 1999. DOI: `10.1109/SFFCS.1999.814600`.

[265] N. Piatkowski, S. Lee, and K. Morik. “Integer undirected graphical models for resource-constrained systems”. In: *Neurocomputing* 173 (2016). DOI: `10.1016/j.neucom.2015.01.091`.

[266] I. Kerenidis, J. Landman, and A. Prakash. “Quantum Algorithms for Deep Convolutional Neural Networks”. In: *Proc. Int. Conf. on Learning Representations (ICLR)*. 2020. URL: `https://arxiv.org/abs/1911.01117v1`.

[267] I. Kerenidis and A. Prakash. “Quantum Gradient Descent for Linear Systems and Least Squares”. In: *Physical Review A* 101.2 (2020). DOI: `10.1103/physreva.101.022316`.

[268] V. Giovannetti, S. Lloyd, and L. Maccone. “Architectures for a Quantum Random Access Memory”. In: *Physical Review A* 78.5 (2008). DOI: `10.1103/physreva.78.052310`.

[269] C. Chamberland et al. “Topological and Subsystem Codes on Low-Degree Graphs with Flag Qubits”. In: *Physical Review X* 10.1 (2020). DOI: `10.1103/PhysRevX.10.011022`. URL: `https://arxiv.org/abs/1907.09528`.

[270] N. Dattani, S. Szalay, and N. Chancellor. “Pegasus: The Second Connectivity Graph for Large-scale Quantum Annealing Hardware”. In: *arXiv:1901.07636 [quant-ph]* (2019). URL: `https://arxiv.org/abs/1901.07636v1`.

[271] R. Dechter and J. Pearl. “The cycle-cutset method for improving search performance in AI applications”. In: *Conference on Artificial Intelligence Applications*. IEEE, 1987.

[272] S. Hadfield et al. “From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz”. In: *Algorithms* 12.2 (2019). ISSN: 1999-4893. DOI: `10.3390/a12020034`.

[273] G. H. Low and I. L. Chuang. “Hamiltonian Simulation by Qubitization”. In: *Quantum* 3 (2019). ISSN: 2521-327X. DOI: `10.22331/q-2019-07-12-163`.

[274] S. Bravyi et al. “On the complexity of quantum partition functions”. In: (2021). arXiv: 2110.15466 [quant-ph].

[275] W. H. Zurek. “Decoherence and the Transition from Quantum to Classical — Revisited”. In: *Quantum Decoherence: Poincaré Seminar 2005*. Ed. by B. Duplantier, J.-M. Raimond, and V. Rivasseau. Birkhäuser Basel, 2007. ISBN: 978-3-7643-7808-0. DOI: 10.1007/978-3-7643-7808-0_1.

[276] B. Vacchini. “Quantum Noise from Reduced Dynamics”. In: *Fluctuation and Noise Letters* 15.3 (2016). ISSN: 1793-6780. DOI: 10.1142/s0219477516400034.

[277] A. M. Childs, E. Farhi, and J. Preskill. “Robustness of adiabatic quantum computation”. In: *Physical Review A* 65.1 (2001). ISSN: 1094-1622. DOI: 10.1103/physreva.65.012322.

[278] P. Q. Le, F. Dong, and K. Hirota. “A flexible representation of quantum images for polynomial preparation, image compression, and processing operations”. In: *Quantum Information Processing* 10.1 (2011). DOI: 10.1007/s11128-010-0177-y.

[279] Y. Zhang et al. “NEQR: a novel enhanced quantum representation of digital images”. In: *Quantum Information Processing* 12.8 (2013). DOI: 10.1007/s11128-013-0567-z.

[280] F. A. González and J. C. Caicedo. “Quantum Latent Semantic Analysis”. In: *Advances in Information Retrieval Theory*. 2011. DOI: 10.1007/978-3-642-23318-0_7.

[281] S. Lloyd, M. Mohseni, and P. Rebentrost. “Quantum principal component analysis”. In: *Nature Physics* 10.9 (2014). DOI: 10.1038/nphys3029.

[282] E. Tang. “Quantum Principal Component Analysis Only Achieves an Exponential Speedup Because of Its State Preparation Assumptions”. In: *Phys. Rev. Lett.* 127 (6 2021). DOI: 10.1103/PhysRevLett.127.060503.

[283] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014. ISBN: 978-1-10-705713-5. URL: http://www.cambridge.org/de/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms.

[284] C. Ciliberto et al. “Statistical limits of supervised quantum learning”. In: *Physical Review A* 102.4 (2020). ISSN: 2469-9934. DOI: 10.1103/physreva.102.042414.

[285] V. Giovannetti, S. Lloyd, and L. Maccone. “Quantum-Enhanced Measurements: Beating the Standard Quantum Limit”. In: *Science* 306.5700 (2004). DOI: 10.1126/science.1104149.

[286] M. B. Hastings. “The Power of Adiabatic Quantum Computation with No Sign Problem”. In: (2020). arXiv: 2005.03791 [quant-ph].

[287] B. Altshuler, H. Krovi, and J. Roland. “Adiabatic quantum optimization fails for random instances of NP-complete problems”. In: (2009). arXiv: 0908.2782 [quant-ph].

[288] B. Altshuler, H. Krovi, and J. Roland. “Anderson localization makes adiabatic quantum optimization fail”. In: *Proceedings of the National Academy of Sciences* 107.28 (2010). DOI: 10.1073/pnas.1002116107.

[289] L. Zhou et al. “Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices”. In: *Physical Review X* 10.2 (2020). ISSN: 2160-3308. DOI: 10.1103/physrevx.10.021067.

[290] J. Guan, W. Fang, and M. Ying. “Robustness Verification of Quantum Classifiers”. In: *Lecture Notes in Computer Science* (2021). ISSN: 1611-3349. DOI: 10.1007/978-3-030-81685-8_7.

[291] B. Weder et al. “The Quantum Software Lifecycle”. In: *Proceedings ACM SIGSOFT Int. W. on Architectures and Paradigms for Engineering Quantum Software (APEQS 2020)*. ACM, 2020. DOI: 10.1145/3412451.3428497. URL: https://www.iaas.uni-stuttgart.de/publications/Weder2020_QuantumSoftwareLifecycle.pdf.

[292] D. Arias et al. “A Repeated Mistake is a Choice: Considering Security Issues and Risks in Quantum Computing from Scratch”. In: *CISIS 2021 and ICEUTE 2021*. Ed. by J. J. Gude Prego et al. Springer International Publishing, 2022.

[293] Federal Office for Information Security. *Sicherer, robuster und nachvollziehbarer Einsatz von KI*. Whitepaper Federal Office for Information Security. 2021. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/KI/Herausforderungen_und_Massnahmen_KI.pdf?__blob=publicationFile&v=6.

[294] Federal Office for Information Security. *AI Cloud Service Compliance Criteria Catalogue (AIC4)*. 2021. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/CloudComputing/AIC4/AI-Cloud-Service-Compliance-Criteria-Catalogue_AIC4.pdf.

[295] A. A. Saki et al. *A Survey and Tutorial on Security and Resilience of Quantum Computing*. 2021. arXiv: 2106.06081. URL: https://arxiv.org/abs/2106.06081.

[296] N. Papernot et al. “Towards the Science of Security and Privacy in Machine Learning”. In: *arXiv cs.CR* abs/1611.03814 (2016). URL: http://arxiv.org/abs/1611.03814.

[297] T. Gabor et al. “The Holy Grail of Quantum Artificial Intelligence: Major Challenges in Accelerating the Machine Learning Pipeline”. In: *arXiv quant-ph* (2020). DOI: 10.48550/arXiv.2004.14035. URL: https://arxiv.org/abs/2004.14035.

[298] C. Berghoff, M. Neu, and A. von Twickel. “Vulnerabilities of Connectionist AI Applications: Evaluation and Defense”. In: *Frontiers in Big Data* (2020). DOI: 10.3389/fdata.2020.00023. URL: https://www.frontiersin.org/article/10.3389/fdata.2020.00023.

[299] V. E. Elfving et al. *How will quantum computers provide an industrially relevant computational advantage in quantum chemistry?* 2020. DOI: 10.48550/ARXIV.2009.12472. URL: https://arxiv.org/abs/2009.12472.

[300] V. Dixit et al. “Training a Quantum Annealing Based Restricted Boltzmann Machine on Cybersecurity Data”. In: *IEEE Trans. on Emerging Topics in Computational Intelligence* (2021). DOI: `10.1109/TETCI.2021.3074916`. URL: `https://arxiv.org/abs/2011.13996v4`.

[301] D. Dong et al. “Quantum Reinforcement Learning”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.5 (2008). DOI: `10.1109/tsmcb.2008.925743`.

[302] Y. Kwak et al. *Introduction to Quantum Reinforcement Learning: Theory and PennyLane-based Implementation*. 2021. DOI: `10.48550/ARXIV.2108.06849`. URL: `https://arxiv.org/abs/2108.06849`.

[303] X. Wang et al. “The security of machine learning in an adversarial setting: A survey”. In: *Journal of Parallel and Distributed Computing* 130 (2019). ISSN: 0743-7315. DOI: `https://doi.org/10.1016/j.jpdc.2019.03.003`. URL: `https://www.sciencedirect.com/science/article/pii/S0743731518309183`.

[304] S. Lu, L.-M. Duan, and D.-L. Deng. “Quantum adversarial machine learning”. In: *Physical Review Research* 2.3 (2020). DOI: `10.1103/physrevresearch.2.033212`.

[305] L. Pinto et al. *Robust Adversarial Reinforcement Learning*. 2017. DOI: `10.48550/ARXIV.1703.02702`. URL: `https://arxiv.org/abs/1703.02702`.

[306] T. Brown et al. “Language Models are Few-Shot Learners”. In: *Proc. NeurIPS*. 2020.

[307] D. Amodei et al. *AI and Compute*. 2018. URL: `https://openai.com/blog/ai-and-compute/`.

[308] J. R. McClean et al. “Barren plateaus in quantum neural network training landscapes”. In: *Nature Communications* 9.1 (2018). DOI: `10.1038/s41467-018-07090-4`.

[309] A. Abbas et al. “The power of quantum neural networks”. In: *Nature Computational Science* (2021). DOI: `10.1038/s43588-021-00084-1`. URL: `https://arxiv.org/abs/2011.00027`.

[310] M. Barreno et al. “Can Machine Learning Be Secure?” In: *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*. ASIACCS ’06. Taipei, Taiwan: Association for Computing Machinery, 2006. ISBN: 1595932720. DOI: `10.1145/1128817.1128824`. URL: `http://www.blaine-nelson.com/research/pubs/Barreno-Nelson-ASIACCS-2006.pdf`.

[311] R. S. S. Kumar et al. *Failure Modes in Machine Learning Systems*. 2019. arXiv: `1911.11034`.

[312] C. Berghoff, M. Neu, and A. von Twickel. “Vulnerabilities of Connectionist AI Applications: Evaluation and Defence”. In: *CoRR* abs/2003.08837 (2020). arXiv: `2003.08837`. URL: `https://arxiv.org/abs/2003.08837`.

[313] B. Biggio and F. Roli. “Wild patterns: Ten years after the rise of adversarial machine learning”. In: *Pattern Recognition* 84 (2018). ISSN: 0031-3203. DOI: `https://doi.org/10.1016/j.patcog.2018.07.023`. URL: `https://www.sciencedirect.com/science/article/pii/S0031320318302565`.

[314] K. He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: `10.48550/ARXIV.1512.03385`. URL: `https://arxiv.org/abs/1512.03385`.

[315] M. Rigaki and S. Garcia. “A Survey of Privacy Attacks in Machine Learning”. In: *CoRR* abs/2007.07646 (2020). arXiv: `2007.07646`. URL: `https://arxiv.org/abs/2007.07646`.

[316] N. Poh. *Tutorial: Applying wolf and hill climbing attacks to fingerprint recognition*. 2017. URL: `https://normanpoh.github.io/blog/2017/12/31/hill-climbing.html`.

[317] D. Ratke. *QUBO NN - Reverse Engineering QUBO matrices*. 2021. URL: `https://blog.xa0.de/post/QUBO-NN%20---%20Reverse-Engineering-QUBO-matrices`.

[318] D. Biesner, R. Sifa, and C. Bauckhage. “Solving Subset Sum Problems using Binary Optimization with Applications in Auditing and Financial Data Analysis”. In: *TechRxiv preprint 18994160.v1* (2022).

[319] Federal Office for Information Security. *Security of AI-Systems: Fundamentals – Adversarial Deep Learning*. in publication. 2022. URL: `https://www.bsi.bund.de/`.

[320] I. J. Goodfellow, J. Shlens, and C. Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: `1412.6572 [stat.ML]`.

[321] X. Ma et al. “Understanding adversarial attacks on deep learning based medical image analysis systems”. In: *Pattern Recognition* 110 (2021). ISSN: 0031-3203. DOI: `10.1016/j.patcog.2020.107332`. URL: `http://dx.doi.org/10.1016/j.patcog.2020.107332`.

[322] S. Sim, P. D. Johnson, and A. Aspuru□Guzik. “Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum□Classical Algorithms”. In: *Advanced Quantum Technologies* 2.12 (2019). DOI: `10.1002/qute.201900070`.

[323] A. Aldahdooh et al. “Adversarial Example Detection for DNN Models: A Review”. In: *CoRR* abs/2105.00203 (2021). arXiv: `2105.00203`. URL: `https://arxiv.org/abs/2105.00203`.

[324] H. Liao et al. “Robust in practice: Adversarial attacks on quantum machine learning”. In: *Physical Review A* 103.4 (2021). ISSN: 2469-9934. DOI: `10.1103/physreva.103.042427`. URL: `http://dx.doi.org/10.1103/PhysRevA.103.042427`.

[325] N. Liu and P. Wittek. “Vulnerability of quantum classification to adversarial perturbations”. In: *Physical Review A* 101.6 (2020). ISSN: 2469-9934. DOI: `10.1103/physreva.101.062331`. URL: `http://dx.doi.org/10.1103/PhysRevA.101.062331`.

[326] W. Gong and D.-L. Deng. “Universal Adversarial Examples and Perturbations for Quantum Classifiers”. In: *National Science Review* (2021). ISSN: 2053-714X. DOI: 10.1093/nsr/nwab130. URL: http://dx.doi.org/10.1093/nsr/nwab130.

[327] ilmoi. *Poisoning attacks on Machine Learning*. 2019. URL: https://towardsdatascience.com/poisoning-attacks-on-machine-learning-1ff247c254db.

[328] IBM Quantum. *IBM Quantum Compute Resources, Calibration data of ibmq_guadalupe*. 2022. URL: https://quantum-computing.ibm.com.

[329] T. Gu et al. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks”. In: *IEEE Access* 7 (2019). DOI: 10.1109/ACCESS.2019.2909068.

[330] S. Goldwasser et al. “Planting Undetectable Backdoors in Machine Learning Models”. In: *arXiv preprint arXiv:2204.06974* (2022). DOI: 10.48550/ARXIV.2204.06974. URL: https://arxiv.org/abs/2204.06974.

[331] C. Zhou and R. C. Paffenroth. “Anomaly Detection with Robust Deep Autoencoders”. In: *Proc. KDD*. 2017.

[332] C. Wang et al. “Poisoning attacks and countermeasures in intelligent networks: Status quo and prospects”. In: *Digital Communications and Networks* (2021). ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2021.07.009. URL: https://www.sciencedirect.com/science/article/pii/S235286482100050X.

[333] R. Shokri, M. Stronati, and V. Shmatikov. “Membership Inference Attacks against Machine Learning Models”. In: *CoRR* abs/1610.05820 (2016). arXiv: 1610.05820. URL: http://arxiv.org/abs/1610.05820.

[334] A. Salem et al. “ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models”. In: *CoRR* abs/1806.01246 (2018). arXiv: 1806.01246. URL: http://arxiv.org/abs/1806.01246.

[335] C. Jiang, C. Xu, and Y. Zhang. “PFLM: Privacy-preserving federated learning with membership proof”. In: *Information Sciences* 576 (2021). ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.05.077. URL: https://www.sciencedirect.com/science/article/pii/S0020025521005703.

[336] C. Jiang. *PFLM*. 2021. URL: https://github.com/JiangChSo/PFLM.

[337] S. Srivastava. *Differential Privacy*. 2021. URL: https://github.com/sastava007/Differential-Privacy.

[338] M. Senekane, M. Mafu, and B. M. Taele. “Privacy-preserving quantum machine learning using differential privacy”. In: *2017 IEEE AFRICON*. 2017. DOI: 10.1109/AFRCON.2017.8095692.

[339] M. Senekane, M. Maseli, and M. B. Taele. "Noisy, Intermediate-Scale Quantum Computing and Industrial Revolution 4.0". In: *The Disruptive Fourth Industrial Revolution: Technology, Society and Beyond*. Ed. by W. Doorsamy, B. S. Paul, and T. Marwala. Cham: Springer International Publishing, 2020. DOI: `10.1007/978-3-030-48230-5_9`.

[340] S. Rummer. *Differentially Private Synthetic Data Generation*. 2022. URL: `https://github.com/stefanrmmr/differentially_private_synthetic_data`.

[341] K. Liu et al. "Privacy-Preserving Multi-task Learning". In: *2018 IEEE International Conference on Data Mining (ICDM)*. 2018. DOI: `10.1109/ICDM.2018.00147`.

[342] N. M. Uplavikar. *PP-Multi-Task-Learning*. 2021. URL: `https://github.com/uplavikarnitish/PP-Multi-Task-Learning`.

[343] H. Poulsen Nautrup et al. "Optimizing Quantum Error Correction Codes with Reinforcement Learning". In: *Quantum* (3 2019). DOI: `10.22331/q-2019-12-16-215`. URL: `https://arxiv.org/abs/1812.08451v5`.

[344] I. Convy et al. "Machine Learning for Continuous Quantum Error Correction on Superconducting Qubits". In: *arXiv:2110.10378 [quant-ph]* (2021). URL: `https://arxiv.org/abs/2110.10378`.

[345] G. Avoine and J. Hernandez-Castro. *Security of Ubiquitous Computing Systems Selected Topics: Selected Topics*. 2021. ISBN: 978-3-030-10590-7. DOI: `10.1007/978-3-030-10591-4`.

[346] H. Maghrebi. "Deep Learning based Side Channel Attacks in Practice". In: *IACR Cryptol. ePrint Arch.* 2019 (2019).

[347] S. Marzougui et al. "Machine-Learning Side-Channel Attacks on the GALACTICS Constant-Time Implementation of BLISS". In: *CoRR* abs/2109.09461 (2021). arXiv: `2109.09461`. URL: `https://arxiv.org/abs/2109.09461`.

[348] D. Das et al. "X-DeepSCA: Cross-Device Deep Learning Side Channel Attack". In: 2019. DOI: `10.1145/3316781.3317934`.

[349] P. Kashyap et al. "2Deep: Enhancing Side-Channel Attacks on Lattice-Based Key-Exchange via 2-D Deep Learning". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40.6 (2021). DOI: `10.1109/TCAD.2020.3038701`.

[350] A. Gohr, S. Jacob, and W. Schindler. "Subsampling and Knowledge Distillation on Adversarial Examples: New Techniques for Deep Learning Based Side Channel Evaluations". In: *International Conference on Selected Areas in Cryptography*. Springer. 2020. DOI: `10.1007/978-3-030-81652-0_22`. URL: `https://eprint.iacr.org/2020/165.pdf`.

[351] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia. "Ultrahigh Error Threshold for Surface Codes with Biased Noise". In: *Physical Review Letters* 120.5 (2018). DOI: `10.1103/physrevlett.120.050505`.

[352] C. Neill et al. “A blueprint for demonstrating quantum supremacy with superconducting qubits”. In: *Science* 360.6385 (2018). DOI: 10.1126/science.aao4309.

[353] V. Dunjko and H. J. Briegel. “Machine learning & artificial intelligence in the quantum domain: a review of recent progress”. In: *Reports on Progress in Physics* 81.7 (2018). DOI: 10.1088/1361-6633/aab406. URL: https://doi.org/10.1088/1361-6633/aab406.

[354] Z. Kong et al. “A Survey on Adversarial Attack in the Age of Artificial Intelligence”. In: *Wireless Communications and Mobile Computing* 2021 (2021). DOI: 10.1155/2021/4907754.

[355] F. Tramèr et al. *Stealing Machine Learning Models via Prediction APIs*. 2016. arXiv: 1609.02943.

[356] M. Fredrikson, S. Jha, and T. Ristenpart. “Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures”. In: *Proc. ACM SIGSAC Conf. on Computer and Communications Security*. CCS ’15. Denver, Colorado, USA: Association for Computing Machinery, 2015. ISBN: 9781450338325. DOI: 10.1145/2810103.2813677. URL: https://doi.org/10.1145/2810103.2813677.

[357] P. M et al. *Leitfaden zur Gestaltung vertrauenswürdiger Künstlicher Intelligenz*. Fraunhofer IAIS. 2021.

[358] W. Song et al. “Quantum secure learning with classical samples”. In: *Physical Review A* 103.4 (2021). ISSN: 2469-9934. DOI: 10.1103/physreva.103.042409. URL: http://dx.doi.org/10.1103/PhysRevA.103.042409.

[359] A. Suresh et al. *A Quantum Circuit Obfuscation Methodology for Security and Privacy*. 2021. DOI: 10.48550/ARXIV.2104.05943. URL: https://arxiv.org/abs/2104.05943.

[360] A. García de la Barrera et al. “Quantum software testing: State of the art”. In: *J. of Software: Evolution and Process* (2021). DOI: https://doi.org/10.1002/smr.2419.

[361] W. M. Watkins, S. Y.-C. Chen, and S. Yoo. *Quantum machine learning with differential privacy*. 2021. arXiv: 2103.06232.

[362] Y.-B. Sheng and L. Zhou. “Distributed secure quantum machine learning”. In: *Science Bulletin* 62.14 (2017). ISSN: 2095-9273. DOI: https://doi.org/10.1016/j.scib.2017.06.007. URL: https://www.sciencedirect.com/science/article/pii/S2095927317303250.

[363] J. Bang, S.-W. Lee, and H. Jeong. “Protocol for secure quantum machine learning at a distant place”. In: *Quantum Information Processing* 14.10 (2015). ISSN: 1573-1332. DOI: 10.1007/s11128-015-1089-7. URL: http://dx.doi.org/10.1007/s11128-015-1089-7.

[364] C. H. Yang et al. “Decentralizing Feature Extraction with Quantum Convolutional Neural Network for Automatic Speech Recognition”. In: *CoRR* abs/2010.13309 (2020). arXiv: 2010.13309. URL: https://arxiv.org/abs/2010.13309.

[365] S. Y.-C. Chen and S. Yoo. *Federated Quantum Machine Learning*. 2021. arXiv: 2103.12010.

[366] X. Zhou and D. Qiu. "Blind quantum machine learning based on quantum circuit model". In: *Quantum Information Processing* 20.11, 363 (2021). DOI: 10.1007/s11128-021-03301-y.

[367] H. Wang et al. "RoQNN: Noise-Aware Training for Robust Quantum Neural Networks". In: *CoRR* abs/2110.11331 (2021). arXiv: 2110.11331. URL: https://arxiv.org/abs/2110.11331.

[368] Z. Katzir and Y. Elovici. "Quantifying the Resilience of Machine Learning Classifiers Used for Cyber Security". In: *Expert Systems with Applications* 92.Feb. (2018). DOI: 10.1016/j.eswa.2017.09.053. URL: https://www.sciencedirect.com/science/article/pii/S0957417417306590.

[369] B. Reese et al. "Predict Better with Less Training Data Using a QNN". In: *arXiv:2206.03960 [quant-ph]* (2022). URL: https://arxiv.org/abs/2206.03960.

[370] Y. Zhao. "Development of Quantum Key Distribution and Attacks against It". In: *J. Phys.: Conf. Ser.* 1087 (4 2018). DOI: 10.1088/1742-6596/1087/4/042028.

[371] C. Pacher et al. "Attacks on quantum key distribution protocols that employ non-ITS authentication". In: *Quantum Information Processing* 15 (2015). DOI: 10.1007/s11128-015-1160-4.

[372] Y. Mao et al. "Detecting quantum attacks: a machine learning based defense strategy for practical continuous-variable quantum key distribution". In: *New Journal of Physics* 22.8 (2020). DOI: 10.1088/1367-2630/aba8d4. URL: https://doi.org/10.1088/1367-2630/aba8d4.

[373] H. A. Al-Mohammed et al. "Machine Learning Techniques for Detecting Attackers During Quantum Key Distribution in IoT Networks With Application to Railway Scenarios". In: *IEEE Access* 9 (2021). DOI: 10.1109/ACCESS.2021.3117405.

[374] W. Liu et al. "Integrating machine learning to achieve an automatic parameter prediction for practical continuous-variable quantum key distribution". In: *Phys. Rev. A* 97 (2 2018). DOI: 10.1103/PhysRevA.97.022316. URL: https://link.aps.org/doi/10.1103/PhysRevA.97.022316.

[375] European Commission. *Press statement by President von der Leyen on the fifth round of sanctions against Russia*. 2022. URL: https://ec.europa.eu/commission/presscorner/detail/en/statement_22_2281.

[376] C. Cimpanu. *US sanctions 28 quantum computing entities in China, Russia, Pakistan, Japan*. 2021. URL: https://therecord.media/us-sanctions-28-quantum-computing-entities-in-china-russia-pakistan-japan/.

[377] D. Bluvstein et al. "A Quantum Processor Based on Coherent Transport of Entangled Atom Arrays". In: *Nature* 604.7906 (2022). DOI: 10.1038/s41586-022-04592-6.

[378] T. Graham and et al. “Multi-qubit Entanglement and Algorithms on a Neutral-Atom Quantum Computer”. In: *Nature* 604.7906 (2022). DOI: 10.1038/s41586-022-04592-6.

[379] L. M. et al. “Quantum Computational Advantage with a Programmable Photonic Processor”. In: *Nature* 606 (2022).

[380] D. Dasgupta, Z. Akhtar, and S. Sen. “Machine Learning in Cybersecurity: A Comprehensive Survey”. In: *J. Defense Modeling and Simulation* 19.1 (2020). DOI: 10.1177/1548512920951275.

[381] B. Naik et al. “The impacts of artificial intelligence techniques in augmentation of cybersecurity: a comprehensive review”. In: *Complex & Intelligent Systems* (2021). DOI: 10.1007/s40747-021-00494-8.

[382] Federal Office for Information Security. *Machine Learning in the Context of Static Application Security Testing – ML-SAST*. 2022. URL: https://www.bsi.bund.de/.

[383] Trend Micro Research, United Nations Interregional Crime and Justice Research Institute (UNICRI), and Europol’s European Cybercrime Centre (EC3). *Malicious Uses and Abuses of Artificial Intelligence*. 2020. URL: https://documents.trendmicro.com/assets/white_papers/wp-malicious-uses-and-abuses-of-artificial-intelligence.pdf.

[384] K. Shaukat et al. “A survey on machine learning techniques for cyber security in the last decade”. In: *IEEE Access* 8 (2020). DOI: 10.1109/ACCESS.2020.3041951. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9277523.

[385] N. Hussain et al. “Spam Review Detection Techniques: A Systematic Literature Review”. In: *Applied Sciences* 9 (2019). ISSN: 2076-3417. DOI: 10.3390/app9050987. URL: https://www.mdpi.com/2076-3417/9/5/987.

[386] M. Crawford et al. “Survey of review spam detection using machine learning techniques”. In: *Journal of Big Data* 2 (2015). ISSN: 2196-1115. DOI: 10.1186/s40537-015-0029-9. URL: https://doi.org/10.1186/s40537-015-0029-9.

[387] L. Lota and B. M. M. Hossain. “A Systematic Literature Review on SMS Spam Detection Techniques”. In: *International Journal of Information Technology and Computer Science* 9 (2017). DOI: 10.5815/ijitcs.2017.07.05.

[388] J. Kremling and A. Parker. *Cyberspace, Cybersecurity, and Cybercrime*. SAGE Publications, 2017. ISBN: 9781506392257. URL: https://books.google.de/books?id=I39ZDwAAQBAJ.

[389] K. Thomas et al. “Trafficking Fraudulent Accounts: The Role of the Underground Market in Twitter Spam and Abuse”. In: *22nd USENIX Security Symposium*. Washington, D.C.: USENIX Association, 2013. ISBN: 978-1-931971-03-4. URL: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/thomas.

[390] R. Y. K. Lau et al. “Text Mining and Probabilistic Language Modeling for Online Review Spam Detection”. In: *ACM Trans. Manage. Inf. Syst.* 2.4 (2012). ISSN: 2158-656X. DOI: 10.1145/2070710.2070716. URL: https://doi.org/10.1145/2070710.2070716.

[391] P. Hayati et al. “Behaviour-Based Web Spambot Detection by Utilising Action Time and Action Frequency”. In: 2010. ISBN: 978-3-642-12164-7. DOI: 10.1007/978-3-642-12165-4_28.

[392] E. G. Dada et al. “Machine learning for email spam filtering: review, approaches and open research problems”. In: *Heliyon* (2019). DOI: 10.1016/j.heliyon.2019.e01802.

[393] S. Shang et al. “A text classification algorithm based on quantum information”. In: 2015. DOI: 10.1109/ICNC.2015.7378020.

[394] D. Liu, X. Yang, and M. Jiang. “A Novel Classifier Based on Quantum Computation”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, 2013. URL: https://aclanthology.org/P13-2086.

[395] J. Shi et al. “Two End-to-End Quantum-inspired Deep Neural Networks for Text Classification”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021). DOI: 10.1109/TKDE.2021.3130598.

[396] D. Baronia. “Hybrid Quantum-Classical Neural Networks for Text Classification”. In: *TechRxiv. Preprint.* (2021). DOI: 10.36227/techrxiv.13488420.v1. URL: https://doi.org/10.36227/techrxiv.13488420.v1.

[397] D. Arthur and P. Date. *A Hybrid Quantum-Classical Neural Network Architecture for Binary Classification*. 2022. DOI: 10.48550/ARXIV.2201.01820. URL: https://arxiv.org/abs/2201.01820.

[398] S. Dörn. “Quantum Algorithms for Graph Traversals and Related Problems”. In: *Proc. of CIE*. 2007. URL: https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.190/Mitarbeiter/doern/GT.pdf.

[399] S. Jeffery and S. Kimmel. “Quantum Algorithms for Graph Connectivity and Formula Evaluation”. In: *Quantum* (2017). DOI: 10.48550/ARXIV.1704.00765. URL: https://arxiv.org/abs/1704.00765.

[400] T. Goto, Q. H. Tran, and K. Nakajima. “Universal Approximation Property of Quantum Machine Learning Models in Quantum-Enhanced Feature Spaces”. In: *Phys. Rev. Lett.* (2021). DOI: 10.1103/PhysRevLett.127.090506. URL: https://arxiv.org/abs/2009.00298.

[401] D. Gibert, C. Mateu, and J. Planes. “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges”. In: *J of Network and Computer Applications* 153 (2020). DOI: 10.1016/j.jnca.2019.102526. URL: https://www.sciencedirect.com/science/article/pii/S1084804519303868.

[402] N. Liu and P. Rebentrost. “Quantum machine learning for quantum anomaly detection”. In: *Phys. Rev. A* (2018). DOI: 10.1103/PhysRevA.97.042315. URL: https://arxiv.org/abs/1710.07405.

[403] O. Lifandali and N. Abghour. “Deep Learning Methods applied to Intrusion Detection: Survey, Taxonomy and Challenges”. In: *2021 Int Conf on Decision Aid Sciences and Application (DASA)*. 2021. DOI: 10.1109/DASA53625.2021.9682357. URL: https://ieeexplore.ieee.org/abstract/document/9682357.

[404] A. Gouveia and M. Correia. “Towards Quantum-Enhanced Machine Learning for Network Intrusion Detection”. In: *IEEE Int Symp on Network Computing and Applications (NCA)*. 2020. DOI: 10.1109/NCA51143.2020.9306691. URL: https://www.gsd.inesc-id.pt/~mpc/pubs/Quantum_NIDS_final.pdf.

[405] H. Suryotrisongko and Y. Musashi. “Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection”. In: *Proc. Computer Science* (2022). DOI: 10.1016/j.procs.2021.12.135. URL: https://www.sciencedirect.com/science/article/pii/S1877050921023590.

[406] E. D. Payares and J. C. Martinez-Santos. “Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview”. In: *Quantum Computing, Communication, and Simulation*. Ed. by P. R. Hemmer and A. L. Migdall. Int. Society for Optics and Photonics. SPIE, 2021. DOI: 10.1117/12.2593297.

[407] T. Zoppi, A. Ceccarelli, and A. Bondavalli. “Unsupervised Algorithms to Detect Zero-Day Attacks: Strategy and Application”. In: *IEEE Access* (2021). DOI: 10.1109/ACCESS.2021.3090957. URL: https://ieeexplore.ieee.org/abstract/document/9461213.

[408] T. T. Nguyen and V. J. Reddi. “Deep reinforcement learning for cyber security”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2019). DOI: 10.1109/TNNLS.2021.3121870. URL: https://arxiv.org/pdf/1906.05799.pdf.

[409] C. Blank et al. “Quantum classifier with tailored quantum kernel”. In: *npj Quantum Information* (2020). DOI: 10.1038/s41534-020-0272-6. URL: https://www.nature.com/articles/s41534-020-0272-6.pdf.

[410] A. Abbas et al. “Effective dimension of machine learning models”. In: *arXiv preprint arXiv:2112.04807* (2021). DOI: 10.48550/ARXIV.2112.04807. URL: https://arxiv.org/abs/2112.04807.

[411] N. V. Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* (2002). DOI: 10.1613/jair.953. URL: https://www.jair.org/index.php/jair/article/download/10302/24590.

[412] M. Chalé and N. D. Bastian. “Challenges and Opportunities for Generative Methods in the Cyber Domain”. In: *Proc. Winter Simulation Conference (WSC)*. IEEE. 2021. DOI: 10.1109/WSC52266.2021.9715504.

[413] V. Dunjko, J. M. Taylor, and H. J. Briegel. “Quantum-Enhanced Machine Learning”. In: *Phys. Rev. Lett.* (2016). DOI: 10.1103/PhysRevLett.117.130501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.117.130501.

[414] K. Sharma et al. “Reformulation of the No-Free-Lunch Theorem for Entangled Datasets”. In: *Physical Review Letters* 128.7 (2022). DOI: 10.1103/physrevlett.128.070501. URL: https://arxiv.org/abs/2007.04900v2.