Capítulo 4

Aprendizado Profundo em Redes Desafiadoras: Conceitos e Aplicações

Kaylani Bochie (UFRJ), Mateus S. Gilbert (UFRJ),
Luana Gantert (UFRJ), Mariana S. M. Barbosa (UFRJ),
Dianne S. V. Medeiros (UFF) e Miguel Elias M. Campista (UFRJ)

Resumo

Este minicurso tem por objetivo apresentar de forma clara e concisa os conceitos técnicos, desafios e aplicações de técnicas de aprendizado profundo nas chamadas Redes Desafiadoras. Essas redes são caracterizadas pelo excesso de dados produzidos e consequente complexidade de operação. Diferente dos minicursos anteriores, este minicurso foca na revisão da literatura relacionada à aplicação de técnicas de aprendizado profundo na solução de problemas complexos das Redes Desafiadoras. Para tanto, este minicurso apresenta e discute os principais conceitos e algoritmos relacionados ao aprendizado de máquina, com foco no aprendizado profundo. Em seguida, a literatura relativa à aplicação desses algoritmos em redes, como redes de Internet das Coisas e de sensores, redes sem fio móveis, redes industriais e redes veiculares é revisada. Este minicurso também apresenta viés experimental, oferecendo um estudo de caso como exercício prático para que os participantes tenham a oportunidade de se debruçar sobre um problema real.

4.1. Introdução

A complexidade crescente dos sistemas computacionais é inegável, dado o aumento vertiginoso na quantidade de dados a serem analisados para a oferta eficiente de serviços. Em redes de computadores, esse fenômeno não poderia ser diferente, sendo consequência direta da escalada no número de usuários, cenários de aplicação, dinâmica dos nós da rede, fontes de dados, etc. Considerar todos esses parâmetros dificulta a análise e manutenção da Qualidade de Serviço (*Quality of Service* – QoS) e o emprego de protocolos e algoritmos clássicos que não possuem a capacidade de aprendizado. Na Internet das Coisas (*Internet of Things* – IoT), por exemplo, a massa de dados proveniente de sensores diversos é um obstáculo adicional. Nas redes sem fio, a dinamicidade da rede

dificulta a distinção entre comportamento esperado e anomalias. Já em redes industriais, um fator crítico é a disponibilidade que deve levar em conta diferentes motivos para falhas e mitigar seus efeitos. Por fim, nas redes veiculares, a previsão da ocorrência e duração de contatos para transferência eficiente de dados é algo que ainda requer esforços em pesquisa, principalmente na ausência de infraestrutura fixa de comunicação. Neste minicurso, as redes caracterizadas pela geração de um grande volume de dados e pela inerente complexidade de operação são chamadas de "*Redes Desafiadoras*".

O conceito de Aprendizado de Máquina (*Machine Learning*) e suas técnicas correlatas existem há décadas [Osherson et al., 1991]. O ressurgimento do interesse sobre essas técnicas foi impulsionado pela contínua evolução do *hardware* e do *software* que permitiu o uso de ferramentas que exigem processamento intensivo. No contexto atual, o aprendizado de máquina aparece como uma ferramenta alternativa aos algoritmos tradicionais, baseados em regras, que possibilita a análise de diferentes cenários em Redes Desafiadoras sem que as regras de análise sejam explicitamente programadas. O aprendizado de máquina consiste em obter uma representação matemática que modele o comportamento de uma função através de um processo chamado de treinamento. No treinamento, a partir de amostras (*samples*), normalmente com múltiplos atributos (*features*), o modelo tem seus parâmetros ajustados de forma a conseguir prever um conjunto inédito de amostras, exercer uma ação de modo autônomo ou extrair informações relevantes. Durante o aprendizado, é possível que existam alvos (*targets*), que as saídas do modelo devem prever. O bom desempenho do modelo depende de forma crucial da escolha de atributos. Assim, a seleção de atributos acaba se tornando uma extensa área de pesquisa no campo de aprendizado de máquina [Mao et al., 2018]. Outro ponto importante é a capacidade do modelo matemático de "compreender" um problema complexo, capacidade que deve vir acompanhada de rápida resposta e baixa sensibilidade a variações.

Tendo em vista a importância da seleção de atributos e a crescente complexidade nos problemas encontrados em Redes Desafiadoras, o Aprendizado Profundo (*Deep Learning*) tem sido revisitado. Pesquisas nessa área datam dos anos 40, passando por três grandes picos de relevância referenciados por outros nomes: *cybernetics* entre as décadas de 40 e 60, *connectionism* entre os anos 80 e 90 e, por fim, aprendizado profundo a partir de meados dos anos 2000 [Goodfellow et al., 2016]. Nos primórdios, a capacidade computacional era um problema crítico, tornando-se um entrave até a segunda fase. Com o rápido desenvolvimento tecnológico na fase atual, o poder computacional deixa de ser um problema tão relevante para a popularização do aprendizado profundo. Como consequência, há um crescente uso das técnicas e dos algoritmos, mesmo aqueles desenvolvidos nas duas fases precedentes, que permeiam as aplicações atuais em diversas áreas de pesquisa. A popularidade desse tipo de aprendizado se deve ao uso de modelos matemáticos mais complexos que melhoram a seleção de atributos e, consequentemente, aprimoram os resultados alcançáveis. Além disso, através da introdução de diversas camadas de processamento, o aprendizado profundo adapta os modelos empregados à dificuldade do problema. Tipicamente, o aprendizado profundo requer modelos que capturem não-linearidades do problema, tornando a modelagem relativamente mais complexa.

Este minicurso tem por objetivo apresentar de forma clara e concisa os conceitos técnicos, desafios e aplicações de técnicas de aprendizado profundo nas chamadas Redes Desafiadoras. Diferentemente dos minicursos apresentados em simpósios brasileiros

anteriores [Comarela et al., 2019, Medeiros et al., 2019], o foco deste minicurso está na revisão não exaustiva da literatura relacionada à aplicação de técnicas de aprendizado profundo em um determinado conjunto de redes com características marcantes. Para isso, primeiramente, o minicurso apresenta e discute os principais conceitos e algoritmos relacionados ao aprendizado de máquina com foco no aprendizado profundo. Em seguida, quatro tipos de **Redes Neurais Artificiais (*Artificial Neural Networks* – ANNs)** são apresentados. O foco nas redes neurais é dado, visto que muitos dos trabalhos em aprendizado profundo aplicam esse tipo de abordagem. Isso é observado através da aplicação desses algoritmos em redes IoT e de sensores, redes sem fio móveis, redes industriais e redes veiculares que, em comum, possuem desafios relacionados majoritariamente à dinamicidade e às limitações de armazenamento e processamento de dados em seus nós. A ideia é capturar, a partir desses trabalhos, abordagens que sejam comuns a um dado tipo de rede ou mesmo a todas como forma de nortear a pesquisa na área. Ao final, uma atividade prática é oferecida aos leitores para fixação do aprendizado adquirido.

Este minicurso está organizado como segue. A Seção 4.2 apresenta os principais conceitos relacionados ao aprendizado de máquina, e principais algoritmos e modelos de aprendizado profundo. A Seção 4.3 aborda aplicações baseadas em aprendizado profundo para solucionar problemas em Redes Desafiadoras. Ainda, essa seção apresenta trabalhos de detecção de intrusão e anomalias, um problema transversal a todas as redes desafiadoras. A Seção 4.4 apresenta ferramentas frequentemente utilizadas na análise de dados de redes de computadores e descreve uma atividade prática focada na detecção de ataques com base em redes neurais profundas. Por fim, a Seção 4.5 conclui este minicurso e apresenta as perspectivas e problemas em aberto relacionados ao tema do minicurso.

4.2. Conceitos de Aprendizado Profundo

Esta seção familiariza o leitor com os termos e técnicas básicas relevantes para a continuidade do minicurso. A seção apresenta conceitos fundamentais de aprendizado de máquina necessários para a compreensão do aprendizado profundo, as principais diferenças entre aprendizado de máquina e profundo e, por fim, os diferentes tipos de redes neurais profundas, que são as principais estruturas utilizadas no aprendizado profundo.

4.2.1. Conceitos Básicos de Aprendizado de Máquina

O aprendizado de máquina surge como um paradigma no qual os algoritmos são capazes de extrair informações dos dados disponibilizados sem serem explicitamente programados [Medeiros et al., 2019]. Dessa forma, é possível reduzir a necessidade de conhecimento prévio sobre o domínio no qual são empregados [Deisenroth et al., 2019]. Essa característica torna as técnicas de aprendizado de máquina atraentes para problemas que envolvem grandes massas de dados, possivelmente de difícil formalização, cuja programação manual de algoritmos tradicionais pode ser custosa ou até mesmo impossível [Goodfellow et al., 2016]. O aumento da popularidade do uso das técnicas de aprendizado de máquina justifica-se pelo casamento entre os cenários atuais, que fazem uso de massas de dados cada vez maiores, e pela evolução na infraestrutura computacional, que oferece o aporte necessário para a execução de técnicas mais avançadas.

O pré-requisito fundamental para os algoritmos que compõem o paradigma do

aprendizado de máquina é a capacidade de *"aprender"*. Mitchell define que um programa está aprendendo se seu desempenho na execução de um conjunto de tarefas, medido a partir de uma métrica de interesse, melhora com a experiência desse conjunto [Mitchell, 1997]. Os algoritmos tiram vantagem de possíveis padrões e estruturas presentes nos dados do problema para ajustar seus parâmetros e, como consequência, melhorar o seu desempenho na tarefa apresentada. Dessa forma, o caráter autônomo dessa abordagem fica evidente, visto que os parâmetros do algoritmo são definidos pela interação do próprio algoritmo com o conjunto de dados. O processo no qual o algoritmo tem seus parâmetros ajustados a fim de aprender a realizar uma tarefa é chamado de *treinamento*, que pode ser feito de diferentes maneiras, dependendo da construção dos dados em relação ao mapeamento entre entradas e saídas. Como consequência, os algoritmos de aprendizado de máquina podem ser divididos em *supervisionado*, *não supervisionado* e *por reforço*.

- **Aprendizado supervisionado**: os algoritmos têm acesso a um conjunto de dados rotulados, ou seja, existem exemplos do mapeamento entre entradas e saídas. A presença dos rótulos possibilita que os algoritmos ajustem seus parâmetros para reproduzirem as mesmas saídas caso entradas semelhantes sejam apresentadas. Em uma analogia ao aprendizado humano, o algoritmo de aprendizado supervisionado tem acesso às respostas corretas das perguntas de um teste e aprende com o acesso a essas respostas. As respostas corretas das perguntas do teste são análogas ao mapeamento entre entradas e saídas promovido pelo rotulamento dos dados.

- **Aprendizado não supervisionado**: o conjunto de dados carece de rótulos, não existindo um mapeamento entre entradas e saídas. Nesse cenário, os algoritmos buscam relações e características presentes no conjunto de dados que possam ser exploradas para classificar internamente os elementos. Essa classificação pode levar a grupos de dados que compartilhem características semelhantes ou a grupos de dados que possuam algum tipo de correlação. As relações inferidas são mensuradas por métricas que verificam se a classificação obtida é adequada, possibilitando que os algoritmos ajustem os seus parâmetros. Analogamente ao aprendizado humano, os algoritmos de aprendizado não supervisionado avaliam padrões, assim como um bebê observa o comportamento e as características que definem uma pessoa conhecida, por exemplo. Observando os padrões de comportamento e características de uma pessoa qualquer, o bebê é capaz de associar aquele conjunto de entradas de dados à saída que determina se a pessoa é conhecida ou não. Não é necessário, nesse caso, que seja informado previamente ao bebê que ele conhece a pessoa.

- **Aprendizado por reforço**: os algoritmos se baseiam em um modelo de recompensas e punições à medida que o modelo interage com o ambiente onde está inserido. Assim, em vez de existir um mapeamento direto entre entradas e saídas, os resultados são obtidos a partir da realimentação (*feedback loop*) entre o sistema de aprendizado e o ambiente. A cada iteração, as ações disponíveis são apresentadas ao modelo no seu estado atual e, após a mudança de estado, recebe um sinal de reforço. De forma similar ao aprendizado humano, os algoritmos de aprendizado por reforço buscam instigar um conjunto de comportamentos desejados, como o uso correto de talheres para uma criança, através de recompensas, como palavras de re-

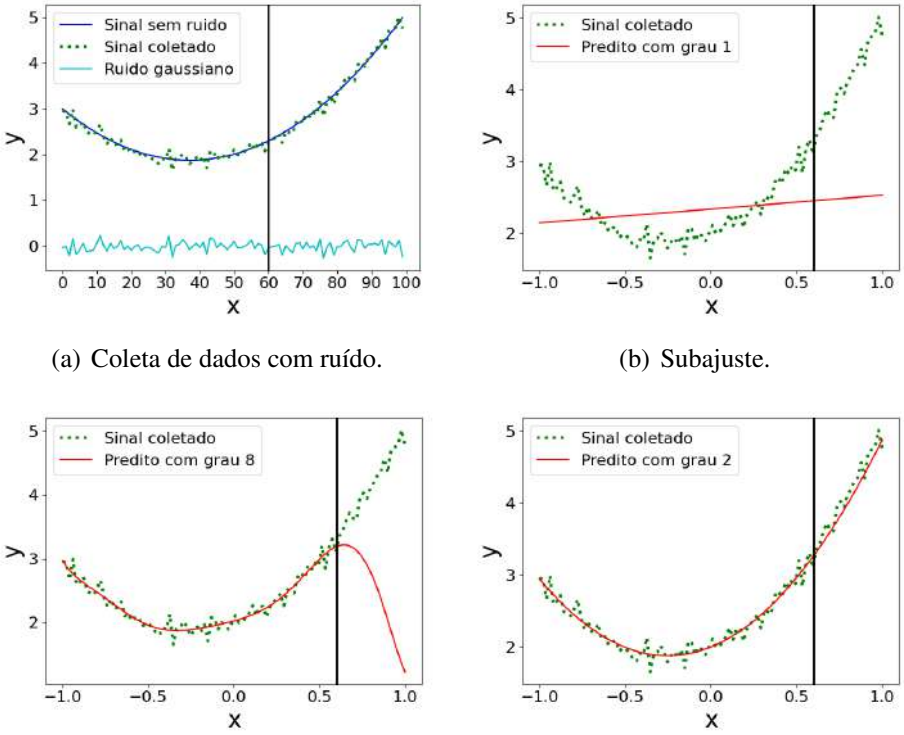
força ou um prêmio. O objetivo dessa abordagem é escolher ações que maximizem a recompensa a longo prazo [Kaelbling et al., 1996].

Ressalta-se que a divisão apresentada é uma forma aproximada de compreender a função dos algoritmos e demonstrar resumidamente o emprego de cada um deles. Porém, outros paradigmas de aprendizado são possíveis. Em especial, alguns desses paradigmas surgem da combinação das classes descritas anteriormente. O **aprendizado semissupervisionado** é um desses casos, que combina elementos dos aprendizados supervisionado e não supervisionado. Esse paradigma é empregado, por exemplo, para rotular conjuntos de dados não rotulados, ou com rótulos faltantes. Para tal, o conjunto de dados é usado como entrada da parte não supervisionada do algoritmo para que este gere os rótulos. Uma vez que todos os dados estão rotulados, a parte supervisionada é usada para classificação.

Além dos tipos de algoritmos, existem conceitos fundamentais que dizem respeito à efetividade de um programa após a modelagem e o próprio aprendizado. Um pré-requisito essencial para o aprendizado de máquina é a capacidade de *generalização*, isto é, o poder de responder de maneira adequada a situações que o algoritmo não teve acesso em seu aprendizado. Para verificar essa capacidade, as amostras dos dados coletados são comumente divididas em três conjuntos: *conjunto de treinamento, conjunto de validação e conjunto de teste*. O **conjunto de treinamento** contém as amostras que o algoritmo efetivamente utiliza para ajustar os seus parâmetros. Portanto, é durante o treinamento que os modelos de aprendizado acessam parte dos dados para ajustar seus parâmetros internos. O processo de treinamento propriamente dito pode diferir de acordo com os modelos de aprendizado adotados. No treinamento, o modelo de aprendizado passa por diversas etapas de atualização de parâmetros e verificação de desempenho. O **conjunto de validação** é usado para acompanhar o progresso no aprendizado, servindo para medir o erro, ou o custo, associado à configuração atual do algoritmo. O modelo utiliza o desempenho atual sobre os dados para alterar seus parâmetros, por exemplo a acurácia, para modelos de classificação, ou o Erro Médio Quadrático (*Mean Squared Error* – MSE) para modelos de regressão. A partir dessa medição, verifica-se se há margem para melhorar o algoritmo ou se o aprendizado deve ser encerrado. O **conjunto de teste** contém amostras inéditas que não foram utilizadas no treinamento e é usado para avaliar o comportamento do modelo já ajustado. Portanto, a avaliação do desempenho do algoritmo é feita a partir do conjunto de teste. É muito importante que o usuário do modelo apenas utilize o conjunto de teste após o ajuste completo do algoritmo feito nos conjuntos de treino e validação. Visto que reajustar características do modelo baseado nos resultados do conjunto de teste provoca **vazamento de dados** (*data leakage*) e provoca a criação de um modelo com desempenho normalmente superior, porque os dados usados durante o treino são usados também para avaliar o desempenho, o que invalida a generalização.

O desempenho de um algoritmo treinado, determinado quanto a sua capacidade de generalização, pode ser caracterizado por dois fatores fundamentais: subajuste (*underfitting*) e sobreajuste (*overfitting*). Durante o treinamento, tanto os casos de subajuste quanto os de sobreajuste devem ser evitados. Os conceitos de subajuste e sobreajuste estão ilustrados na Figura 4.1. A Figura 4.1(a) mostra um conjunto de dados fictício usado para treinar um algoritmo de aprendizado para regressão. Uma coleta de dados naturalmente é contaminada por ruído. Assim, os dados estudados representam um polinômio de

grau 2 com adição de ruído gaussiano, conforme ilustrado pelos pontos verdes. Durante o treinamento, o algoritmo tem acesso aos pontos coletados à esquerda da linha vertical preta e os pontos à direita dessa linha são as amostras inéditas usadas durante o teste.



(a) Coleta de dados com ruído.

(b) Subajuste.

(c) Sobreajuste.

(d) Complexidade apropriada.

Figura 4.1. Um algoritmo de regressão utilizado para aproximar (a) uma curva polinomial de grau 2 com ruído aditivo, pode apresentar subajuste ou sobreajuste. Os dados à esquerda da linha vertical são usados para treinamento, e o restante para teste. (b) No subajuste, o modelo não é parametrizado de forma adequada e prediz dados de teste com baixa acurácia. (c) No sobreajuste, o modelo é bem ajustado aos dados de treinamento, mas não é capaz de predizer dados inéditos. (d) Um modelo de complexidade apropriada apresenta baixo erro no treinamento e prediz dados inéditos com elevada acurácia.

O **subajuste** ocorre quando o algoritmo não está ajustado para executar a tarefa adequadamente, apresentando um elevado erro no treinamento. Nessa situação, o algoritmo falha na execução da tarefa designada pois o modelo definido por seus parâmetros não está ajustado apropriadamente. Um exemplo de subajuste é um algoritmo ajustado para aproximar pontos de um polinômio de ordem 2 por uma reta, como pode ser visto na Figura 4.1(b). Nessa situação é desejável continuar o treinamento, pois a experiência obtida das amostras de treino é benéfica para a melhora do algoritmo. Como é de se esperar, o erro ao longo do treinamento tende assintoticamente a um valor mínimo. Quanto maior a exposição do algoritmo às amostras de treinamento, melhor é o resultado obtido ao tratá-las. Porém, existe um momento no qual o treinamento excessivo do modelo culmina em um pior desempenho para amostras inéditas. Esse efeito é conhecido como **sobreajuste**. Quando isso ocorre, pode-se dizer que o algoritmo se especializa de tal forma que aprende as particularidades do conjunto de amostras utilizado durante o treinamento. Isso

faz com que o modelo apresente um desempenho ruim quando aplicado em amostras que diferem, mesmo que minimamente, do conjunto usado no treinamento. Na Figura 4.1(c) é possível observar que o modelo é ajustado de tal forma que as múltiplas amostras de um polinômio de grau 2, que possui flutuações naturais durante a coleta, são descritas através de um polinômio de grau 8 que apresenta um melhor ajuste às amostras de treino. Nota-se que tanto no sobreajuste (Figura 4.1(c)) quanto no subajuste (Figura 4.1(b)) o algoritmo falha em prever o valor das funções ao ser apresentado a novas amostras. Visto que ambos os casos são prejudiciais ao desempenho do algoritmo, o objetivo do treinamento é encontrar o ponto ótimo que maximiza o desempenho do modelo em amostras inéditas. A Figura 4.1(d) apresenta um modelo com complexidade apropriada para a predição de novas amostras no caso do regressor polinomial.

As técnicas de *regularização* podem ser usadas para que um algoritmo tenha maior margem para minimizar o erro de aprendizado sem que ocorra uma situação de sobreajuste. A função dos regularizadores é aumentar a capacidade de generalização do algoritmo, ou seja, fazer com que o desempenho do modelo no conjunto de teste aumente com o tempo de treinamento. Um exemplo de regularizador, no contexto de um regressor polinomial, é o acréscimo de um termo à função custo que controla a magnitude dos coeficientes do polinômio, impedindo que eles assumam valores muito altos.

Como a saída do modelo de aprendizado é uma função de seus parâmetros internos e dos valores inseridos, e as entradas não podem ser alteradas pelo algoritmo, o ajuste dos parâmetros deve ser feito para criar a função de transferência correta. Dessa forma, torna-se importante distinguir *parâmetros* de *hiperparâmetros*, que fazem parte do processo de aprendizado. Os **parâmetros** são valores aos quais o algoritmo tem acesso e é capaz de alterar, isto é, são valores que podem ser ajustados pelo algoritmo durante o treinamento. Já os **hiperparâmetros** são externos ao programa, sendo definidos *a priori* com a função de auxiliar o aprendizado, como o grau de um polinômio em um regressor polinomial. Os hiperparâmetros podem ser determinados pelo desenvolvedor ou com o auxílio de outro algoritmo de aprendizado de máquina, contanto que não possam ser ajustados pelo programa original. Em suma, as variáveis que o algoritmo pode alterar são chamadas de parâmetros, enquanto aquelas que ele não tem acesso são os hiperparâmetros. A escolha e o ajuste de hiperparâmetros devem ser feitos com o objetivo de maximizar o desempenho do modelo em uma tarefa real. É comum que os hiperparâmetros, quando ajustados de maneira manual, sejam determinados através de tentativa e erro. Porém, existem heurísticas para a escolha dos hiperparâmetros em função do problema abordado e do modelo escolhido [Wu et al., 2019, Neary, 2018].

4.2.2. Aprendizado Profundo: Um Caso de Aprendizado de Máquina

No aprendizado de máquina tradicional, alguns problemas requerem pré-tratamento dos dados que, por sua vez, exigem conhecimento prévio do desenvolvedor [LeCun et al., 2015]. Um exemplo é o desenvolvimento de uma aplicação que mapeia dados não lineares em uma representação linear, necessária para alguns algoritmos de aprendizado de máquina. Uma forma de contornar essa necessidade é fazer com que o algoritmo aprenda não só a mapear os dados de entrada tratados para a saída, como também a representação necessária para viabilizar tal mapeamento. Além disso, outro problema comum é encontrado quando o aprendizado de tarefas complexas ou conceitos abstratos são necessários,

onde o aprendizado da representação é quase tão difícil quanto o aprendizado da função em si [Goodfellow et al., 2016]. Nessa direção, o aprendizado profundo é uma área do aprendizado de máquina caracterizada pela extração dessas representações complexas a partir de representações mais simples. As soluções mais simples são organizadas em uma hierarquia cujos diferentes níveis se complementam para a composição de informações complexas [Goodfellow et al., 2016].

A divisão de um problema complexo em problemas menores permite que porções do algoritmo se especializem em tarefas mais simples, que depois são recombinadas em etapas posteriores para viabilizar a resolução da tarefa mais complexa. No reconhecimento de imagens, por exemplo, um algoritmo pode ser subdividido em módulos responsáveis pela identificação de determinadas partes de uma imagem, como as bordas. Posteriormente os módulos de identificação de borda podem ser combinados para identificar molduras e assim sucessivamente, até que a combinação dos módulos esteja especializada o suficiente para identificar uma figura. Assim, o modelo de aprendizado é dividido em múltiplas camadas de processamento. As camadas são organizações paralelas das estruturas de processamento básicas, que permitem a extração de informações progressivamente mais complexas à medida em que os dados são processados através das múltiplas camadas intermediárias, de forma hierárquica. Apesar de a definição de aprendizado profundo englobar quaisquer modelos de aprendizado que seguem essa abordagem hierárquica de processamento, o termo é comumente utilizado para fazer referência às **Redes Neurais Profundas** (*Deep Neural Networks* – DNNs). Existem outras abordagens aplicadas às Redes Desafiadoras, como o Aprendizado Profundo por Reforço (*Deep Reinforcement Learning* – DRL), que une redes neurais profundas a arquiteturas de aprendizado por reforço. No entanto, este minicurso foca nos diversos tipos existentes de DNN.

4.2.3. Redes Neurais Profundas

O aprendizado profundo engloba estruturas diversas que oferecem múltiplas camadas de processamento, porém as **redes neurais** concentram a maior parte dos esforços de pesquisa e dos algoritmos utilizados. Surgindo originalmente como uma tentativa de simular estruturas neuronais biológicas, a rede neural é uma estrutura composta por células computacionais simples, chamadas **neurônios**, massivamente interconectadas em uma estrutura paralelamente distribuída capaz de armazenar e processar informações para exercer uma tarefa [Haykin, 1994].

A estrutura típica de um neurônio pode ser observada na Figura 4.2. Tratando a entrada como um vetor $\boldsymbol{x}$, suas componentes $x_1, x_2 \ldots x_m$ correspondem às entradas do neurônio que são combinadas em uma soma ponderada, na qual esses valores são escalados utilizando seus respectivos pesos, representados por w_{k1}, w_{k2} ... w_{kn}. O limiar de ativação inerente (*bias*) é introduzido como um grau de liberdade adicional para manipular a saída do neurônio, permitindo ao neurônio um melhor ajuste. O resultado da soma entre o limiar de ativação e as entradas multiplicadas pelos **pesos** produz o potencial de ativação υ_k. A saída, normalmente indicada pelo vetor y_k, corresponde ao potencial de ativação transformado pela função de ativação $\varphi(.)$. A função $\varphi(.)$ é normalmente uma função não-linear do tipo $f(x) = max(x, 0)$ ou uma função do tipo **softmax**, que mapeia a saída em um intervalo entre 0 e 1.

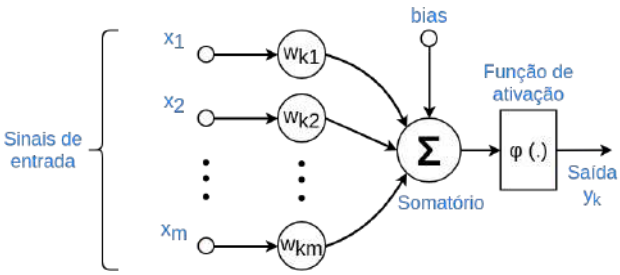


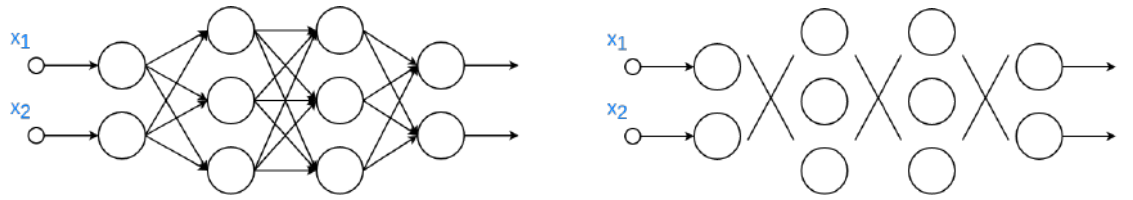
Figura 4.2. Estrutura básica de um neurônio. O vetor de entrada é multiplicado por pesos. Os elementos ponderados e um limiar de ativação (*bias*) são somados e o potencial de ativação resultante alimenta uma função de ativação. A saída é o resultado da transformação do potencial de ativação.

Nas redes neurais, o aprendizado ocorre com o ajuste dos pesos (valores das conexões) entre os neurônios e dos limiares de ativação (*biases*). Esses parâmetros são alterados durante a fase de treinamento. Os pesos são utilizados para calcular a taxa de crescimento da função que o algoritmo tenta modelar, enquanto os limiares de ativação são necessários para deslocar a saída da função. Uma função linear do tipo $y = \boldsymbol{\omega}^T \times \boldsymbol{x} + b$ pode ser modelada escolhendo apropriadamente os pesos ($\boldsymbol{\omega}$) e o limiar de ativação (b), por exemplo. Para cada amostra utilizada no treinamento, o modelo calcula o valor da saída para os valores atuais dos pesos e limiares de ativação, e compara o resultado com o valor esperado (alvo) da amostra. Uma função custo (*cost function*), calculada a partir de uma função de perda (*loss function*) aplicada a diversas amostras, é utilizada para quantificar o erro encontrado para a configuração atual do modelo e um **vetor gradiente**. Isto é, um vetor que contém a variação da saída do modelo em relação a cada um de seus parâmetros é computado para que os pesos sejam atualizados. O modelo atualiza os pesos e limiares de ativação no sentido contrário do vetor gradiente, buscando minimizar a função custo de acordo com uma taxa fixa ou passo, a taxa de aprendizado (*learning rate*).

Os neurônios nas redes neurais são normalmente organizados em grupos de unidades de processamento chamados de **camadas**, que por sua vez são organizadas em uma cadeia. A primeira e a última camada são chamadas de camada de entrada e saída, enquanto as demais são as camadas ocultas (ou internas). As redes neurais profundas se caracterizam pela composição de múltiplas camadas ocultas, como ilustrado na Figura 4.3(a). A Figura 4.3(b) apresenta uma representação simplificada e equivalente à Figura 4.3(a), normalmente utilizada para fins de ilustração. No aprendizado profundo, o aumento no número de camadas ocultas permite que cada uma delas se especialize em identificar um conjunto de características em particular. As camadas mais próximas da camada de entrada são responsáveis por identificar as características mais primitivas, como arestas em uma imagem, enquanto as seguintes combinam essas informações para identificar padrões mais complexos, como animais ou semáforos na mesma imagem. São diversos os tipos de redes neurais existentes. Neste minicurso, são discutidas as **Redes Neurais sem Realimentação**, **Convolucionais**, **Recorrentes** e **Autoassociativas**.

Redes Neurais sem Realimentação

Nas Redes Neurais sem Realimentação (*Feedforward Neural Networks*), a informação é passada entre camadas em apenas um sentido a partir da entrada. O objetivo



(a) Representação tradicional de uma rede neural com duas camadas ocultas.

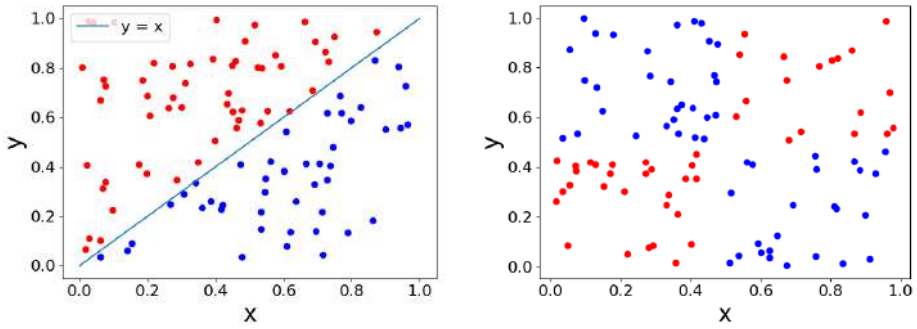
(b) Representação simplificada equivalente de uma rede neural com duas camadas ocultas.

Figura 4.3. Um exemplo de rede neural com duas camadas ocultas, representada de forma (a) tradicional e (b) simplificada. Na simplificação, a ligação entre camadas adjacentes é substituída por um "X".

desse tipo de rede é aproximar uma função qualquer, definindo um mapeamento do tipo $\boldsymbol{y} = f(\boldsymbol{x};\boldsymbol{\theta})$, onde a entrada $\boldsymbol{x}$ é conhecida e os parâmetros $\boldsymbol{\theta}$ são ajustados para oferecer a melhor aproximação da função desejada [Goodfellow et al., 2016]. A rede age, então, como uma composição de funções, na qual cada camada representa uma função da composição. Um dos exemplos mais conhecidos dessas redes é o **Perceptron de Múltiplas Camadas** (*Multilayer Perceptron* – MLP), que usualmente é construído como uma rede totalmente conectada (*fully connected network*) na qual todos os neurônios de uma camada são conectados aos neurônios da camada seguinte.

Uma camada da rede neural sem realimentação pode ser descrita através de uma função de transferência do tipo $\boldsymbol{h}_i = \varphi_i(\boldsymbol{W}_i^T \boldsymbol{h}_{i-1} + \boldsymbol{b}_i)$, onde $\boldsymbol{h}_i$, $\boldsymbol{h}_{i-1}$, $\boldsymbol{b}_i$ e φ_i são, respectivamente, a saída da camada atual, a saída da camada anterior ou entrada da camada atual, o limiar de ativação da camada atual e a função de ativação de uma dada camada. O termo $\boldsymbol{W}_i^T$ é a matriz transposta que contém os pesos entre a camada anterior e a atual. Como esperado, nos casos das camadas de entrada e saída, a entrada da primeira é $\boldsymbol{h}_0 = \boldsymbol{x}$ e a saída da última, $\boldsymbol{h}_n = \boldsymbol{y}$. Nas camadas ocultas, a função de ativação φ_i é usualmente não-linear, porém na última camada, a função pode não conter essa propriedade. O uso de funções não-lineares nas camadas internas é o que confere à rede uma maior capacidade de representação. A necessidade da função φ_i ser não-linear é consequência da incapacidade da rede sem realimentação como um todo ser não-linear, caso todas as funções φ_i fossem lineares. Nesse caso, a rede neural não seria capaz de resolver um problema que não é linearmente separável. Um exemplo é a incapacidade de modelos lineares em representar a função *XOR*. É possível observar na Figura 4.4(a) que um modelo linear é capaz de separar todos os valores da saída da função utilizando apenas uma reta, em contraste com o problema representado na Figura 4.4(b), que não é linearmente separável.

Uma rede neural sem realimentação é um aproximador universal. Isso quer dizer que essas redes têm a capacidade de representar qualquer função, dado que a rede neural seja suficientemente complexa [Goodfellow et al., 2016]. Trabalhos provam que uma rede neural sem realimentação clássica com apenas uma camada oculta é capaz de representar qualquer função mensurável, garantindo que a rede possua quantidade suficiente de neurônios na camada oculta [Cybenko, 1989, Hornik et al., 1989]. Porém, a literatura relata que para problemas complexos, o número de neurônios em apenas uma camada oculta é proibitivamente grande, o que torna a implementação de redes com essa arquitetura impraticável. Por outro lado, o acréscimo de mais camadas ocultas permite explorar



(a) Distribuição de dados linearmente separável.

(b) Distribuição de dados tipo $x \oplus y$, não linearmente separável.

Figura 4.4. (a) Os dados em análise podem apresentar uma distribuição que permita uma separação linear. (b) Contudo, existem dados que não são linearmente separáveis, como distribuições do tipo "XOR".

redes neurais sem realimentação com um número menor de neurônios por camada [Goodfellow et al., 2016, Liang e Srikant, 2016]. Mesmo assim, deve haver uma ponderação sobre o número de camadas utilizadas, já que redes profundas são mais difíceis de otimizar em relação a redes rasas, devido ao número elevado de parâmetros, o que sugere que essa escolha de projeto seja feita via experimentação [Goodfellow et al., 2016].

Redes Neurais Convolucionais

As redes neurais tradicionais, como o MLP, podem não ser indicadas para problemas com dados que apresentam muitas variáveis, além de serem ineficientes no tratamento de dados que possuam algum tipo de estrutura inerente, como relações espaciais. Para tais situações pode ser necessário um número elevado de pesos a serem ajustados, já que um neurônio se conecta às entradas de todos os neurônios da camada seguinte. Isso pode provocar um aumento no tempo de treinamento, além de exigir um número muito elevado de amostras para ajustar os parâmetros corretamente. Quando a segunda situação não é satisfeita, a rede não só não é capaz de aprender algumas variações relevantes desses dados como também fica sujeita a sobreajuste. Além disso, a estrutura de alguns dados como imagens, por exemplo, pode resultar em regiões da rede ajustadas para valores iguais [LeCun et al., 1995], o que além de um treinamento desnecessário pode causar ineficiência no armazenamento dos parâmetros da rede. As Redes Neurais Convolucionais (*Convolutional Neural Networks* – *ConvNets* – CNNs) definem estruturas que contornam tais problemas e, consequentemente, a sua aplicação já pode ser considerada uma tendência em redes de computadores. A adoção em redes é simplificada pela existência de algumas técnicas que convertem os dados provenientes das redes de computadores em formatos apropriados para o uso em CNNs [Sharma et al., 2019].

Duas características das redes neurais convolucionais se destacam: menor número de parâmetros e resistência a variações nos dados, como translação e distorção. A esparsidade de conexões entre os neurônios de camadas adjacentes e o compartilhamento de parâmetros são fundamentais para tornar mais eficiente a retirada de características locais dos dados. Dessa forma, torna-se mais difícil que apenas pedaços da rede neural se espe-

cialize em uma tarefa enquanto outros pedaços são subutilizados. A esparsidade é obtida limitando o número de conexões que um neurônio pode fazer, isto é, cada neurônio recebe apenas saídas de uma pequena vizinhança de neurônios da camada anterior [Goodfellow et al., 2016]. Um exemplo em processamento de imagens é permitir que apenas unidades de *pixels* próximos se conectem a um neurônio de uma camada posterior.

A restrição de conectividade faz com que o conjunto de neurônios passe a se especializar em identificar características primitivas dos dados analisados. Em processamento de imagens as características primitivas podem ser arestas ou contornos, por exemplo. Como essas características estão presentes em diversas partes dos dados analisados, ajustar os parâmetros para cada região implica gastos desnecessários para gerar valores similares. O compartilhamento de pesos permite que apenas um conjunto de pesos seja determinado e seja replicado para as demais regiões da camada. Na prática, apenas uma grade de valores discretos é determinada, chamada de filtro ou **Núcleo Convolucional** (*Convolutional Kernel*) [Khan et al., 2018], onde a estrutura é "passada" por todo o dado analisado. Como esperado, o tamanho dos núcleos convolucionais é menor do que a implementação das camadas como um todo, o que garante maior economia de memória e melhora na eficiência estatística [Goodfellow et al., 2016]. Os filtros convolucionais computam o produto escalar (*dot product*) entre os valores da camada anterior e os valores do filtro. O valor de cada célula dos filtros convolucionais deve ser aprendido pela rede. A representação de uma operação do filtro convolucional pode ser observada na Figura 4.5. O valor 47 presente na saída da operação é obtido a partir do produto escalar entre os elementos sombreados na matriz de entrada, região em que $i, j \leq 3$, e o filtro convolucional. O produto escalar é feito também sobre todas as submatrizes da matriz de entrada e o filtro para obter a saída completa.

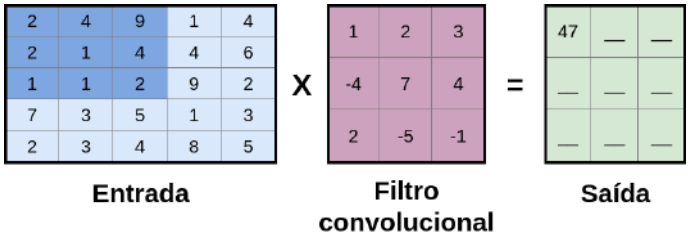


Figura 4.5. Uma operação do filtro convolucional 3×3. A saída é obtida a partir do produto interno entre uma submatriz da matriz de entrada e o filtro convolucional. No exemplo, a submatriz é composta pelos elementos da região em que $i, j \leq 3$, formando uma matriz 3×3.

Uma camada convolucional é tipicamente formada por três estágios. O primeiro deles, chamado de *estágio convolucional*, produz saídas lineares a partir das convoluções necessárias. O segundo estágio, chamado *estágio de detecção*, é responsável por aplicar uma função de ativação não-linear sobre os valores gerados. A saída desses estágios é chamada de mapeamento de características. O terceiro, chamado de *estágio de pooling*, opera sobre o mapeamento das características para combinar os valores em dadas regiões. Uma arquitetura típica de uma rede neural convolucional pode ser observada na Figura 4.6

As redes neurais convolucionais obtêm sua invariância quanto a pequenas translações e distorções através da função de *pooling*, como *average pooling* e *max pooling*.

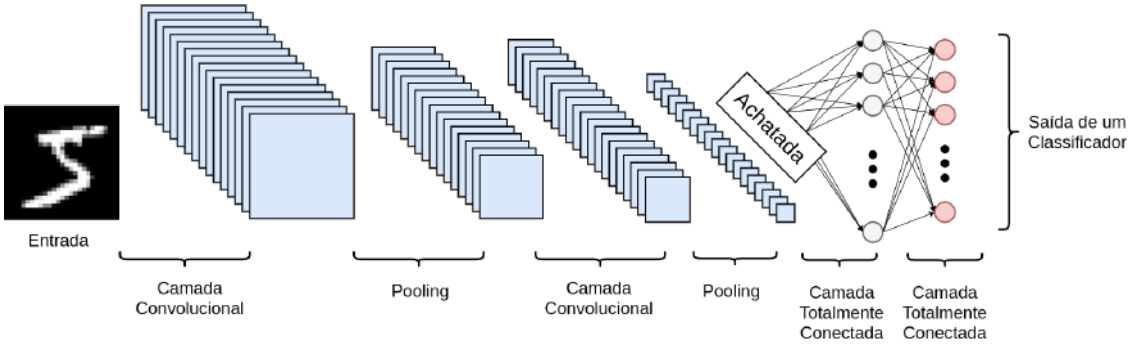


Figura 4.6. Arquitetura típica de uma rede neural convolucional com camadas convolucionais, camadas de *pooling* e camadas totalmente conectadas.

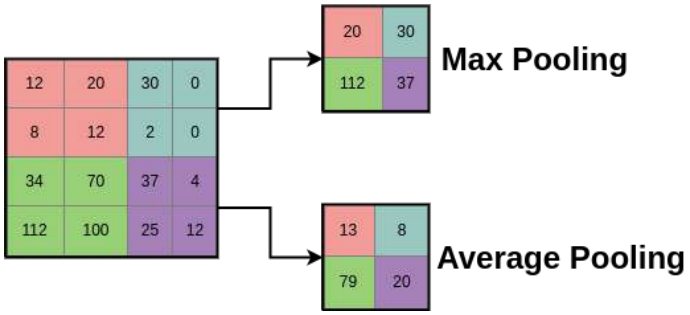


Figura 4.7. Operações de *max pooling* e *average pooling*. Enquanto *average pooling* resulta em uma média das saídas, *max pooling* considera apenas o maior valor de uma vizinhança.

Essas operações suavizam ou até neutralizam as variações. A operação de *average pooling* calcula a média das submatrizes de entrada, enquanto a operação de *max pooling* considera apenas o maior valor de uma dada submatriz. As duas operações estão ilustradas na Figura 4.7. Outra vantagem do uso da camada de *pooling* é a possibilidade de reduzir o número de dimensões do dado tratado, uma vez que essa camada faz uso das saídas de neurônios em uma dada vizinhança. Já a resistência das redes neurais convolucionais em relação a variações nos dados é suportada pelo uso de preenchimento por zeros (*zero padding*). Essa técnica inclui zeros nos dados de entrada da camada convolucional, aumentando a robustez da rede a dados de tamanhos variados. Isso faz com que as saídas das camadas convolucionais assumam o tamanho desejado, viabilizando o funcionamento geral da rede. Sem esse artifício o poder da rede convolucional é comprometido, visto que o desenvolvedor é forçado a escolher uma rápida diminuição na extensão espacial da rede ou o uso de núcleos convolucionais menores [Goodfellow et al., 2016].

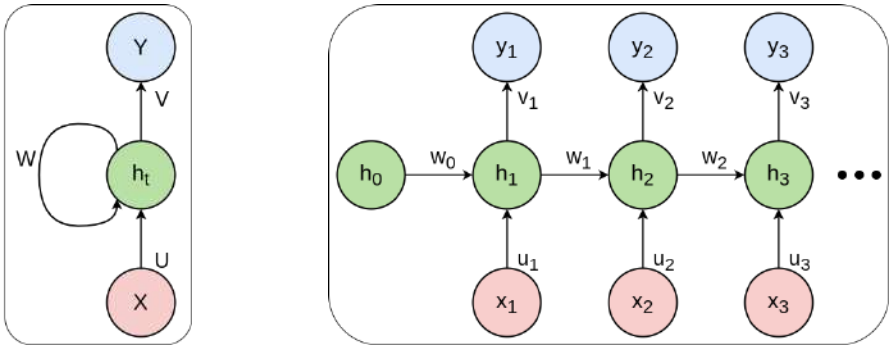
Redes Neurais Recorrentes

Nas redes neurais sem realimentação a informação flui em um único sentido, da entrada para a saída. Essa característica torna o emprego dessas redes muito trabalhoso em cenários de tratamento de dados sequenciais. O relaxamento dessa característica, permitindo a existência de conexões cíclicas, resulta nas Redes Neurais Recorrentes (*Recurrent Neural Networks* – RNNs), ou apenas Redes Recorrentes [Graves, 2012b]. Através das conexões recorrentes, a rede tem acesso a estados anteriores, isto é, valores inseridos e

computados em iterações passadas. Isso permite que RNNs mapeiem qualquer sequência em outra sequência dado um número suficiente de unidades ocultas, conferindo um resultado equivalente à aproximação universal visto nas redes neurais sem realimentação [Hammer, 2000].

A introdução de recorrências permite que as RNNs compartilhem parâmetros entre estados através de diferentes iterações, o que permite que a rede processe sequências de tamanhos variados nas quais as informações de interesse podem ocorrer em posições variadas [Goodfellow et al., 2016]. Um exemplo são as sentenças que podem ser escritas em ordens distintas, representando o mesmo sentido. Uma rede neural sem realimentação precisa ter acesso a todas as diferentes ocorrências da sentença, enquanto uma RNN não tem essa necessidade. De modo geral, as RNNs compõem uma família de redes neurais especializada em processar dados sequenciais [Goodfellow et al., 2016]. Em um ambiente de redes de computadores, uma RNN possui maior potencial para identificar ataques que possuam características temporais ou sequenciais, como os encontrados em Ataques de Negação de Serviço (*Denial of Service* – DoS).

O compartilhamento de estados requer a introdução de novos pesos à rede para possibilitar que a saída da rede seja calculada em função das entradas, do estado da rede e das saídas anteriores. Além da diferença na computação dos resultados, também é importante mencionar que, em contraste com as entradas de tamanho fixo nas redes neurais vistas anteriormente, a entrada de uma RNN é feita de forma gradual e não possui limitação quanto ao tamanho.



(a) Representação "dobrada" de uma rede neural recorrente.

(b) Representação "desdobrada" de uma rede neural recorrente.

Figura 4.8. Representação equivalentes de uma rede neural recorrente.

A computação feita por uma RNN típica é dada por $\boldsymbol{h}_t = \varphi_h(\boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b})$ para a obtenção do novo estado e $\boldsymbol{y}_t = \varphi_y(\boldsymbol{V}\boldsymbol{h}_t + \boldsymbol{c})$ para a obtenção da saída atual, onde $\boldsymbol{b}$ e $\boldsymbol{c}$ são limiares de ativação, $\boldsymbol{U}$, $\boldsymbol{V}$ e $\boldsymbol{W}$ são, respectivamente, as matrizes de pesos das conexões da entrada para a camada oculta, da camada oculta para a saída e da camada oculta para a própria camada oculta, $\boldsymbol{h}_t$ corresponde ao estado atual, e $\boldsymbol{y}_t$ indica a saída da camada. O índice t diz respeito à iteração no tratamento da sequência. Por fim, φ_h e φ_y são funções de ativação não-lineares. Para ilustrar as computações feitas por uma rede recorrente é comum o uso da representação desdobrada, conforme Figura 4.8(b), sendo equivalente a

representação vista na Figura 4.8(a). A representação desdobrada denota os estados da recursão como nós de uma rede, permitindo assim a análise da rede recorrente como se a rede fosse acíclica. A figura evidencia que a computação na direção da entrada para a saída (*forward*) é similar à computação executada em uma rede sem realimentação, com o acréscimo dos pesos referentes à realimentação. Outro ponto em que as duas operações se diferem é o fato de nas redes recorrentes existir a necessidade de armazenar os valores intermediários referentes a um ciclo de amostras da sequência tratável pela rede.

Uma limitação comum das redes neurais recorrentes é a dificuldade no aprendizado de dependências de longo prazo. Ou seja, as RNNs tradicionais têm dificuldade de computar a influência de estados muito distantes da iteração atual. Durante o aprendizado em redes que usam sequências longas, os gradientes tendem a desaparecer ou explodir, fenômenos chamados de *vanishing gradient* e *exploding gradient* respectivamente, inviabilizando a capacidade da rede de aprender as dependências de longo prazo [Bengio et al., 1994]. Uma forma de superar esse problema é utilizar uma variação de RNNs conhecida como Redes Recorrentes com Portas (*Gated Recurrent Networks*).

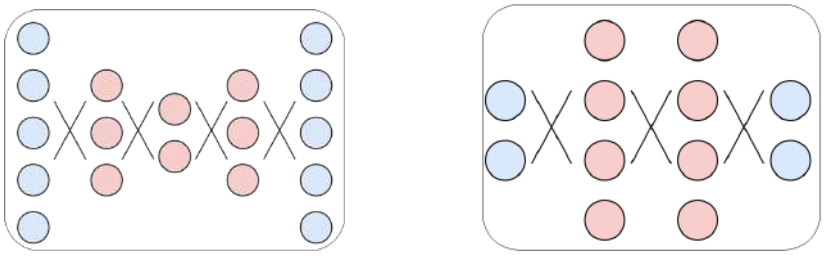
Uma das arquiteturas mais utilizadas em redes recorrentes com portas é a ***Long Short-Term Memory Network* – LSTM**. Nessa variação, as unidades computacionais da camada oculta são substituídas por uma estrutura chamada *bloco de memória*. Cada bloco é composto por uma célula de memória responsável por armazenar as informações a serem mantidas a longo prazo e pelas portas de entrada, de saída e de esquecimento (*forget gate*). As três portas são unidades de soma não-lineares, que controlam o uso da memória [Graves, 2012a]. A porta de esquecimento, quando ativada, é responsável pelo armazenamento do estado atual da unidade. As portas de entrada e saída são responsáveis pela escrita da célula de memória e pela disponibilidade da informação armazenada, respectivamente. A célula "lembra", então, a entrada de um dado instante enquanto as portas de entrada estiverem "fechadas" e a de esquecimento "aberta" [Graves, 2012a].

Em situações em que se deseja analisar uma sequência como um todo, é comum usar a arquitetura de rede chamada **Rede Recorrente Bidirecional**. Essa rede é uma composição de duas redes recorrentes unidirecionais (tradicionais) que analisam a sequência em sentidos opostos. É evidente o caráter não-causal dessas redes, indicando que essas arquiteturas só podem ser empregadas em situações onde a não-causalidade é suportada [Graves, 2012b]. Essa rede pode ser aplicada em problemas que envolvem reconhecimento de fala, nos quais a interpretação de um som em um dado estado pode ser influenciada por fonemas ou palavras que seguem ou precedem [Goodfellow et al., 2016].

Redes Neurais Autoassociativas

O objetivo da Rede Neural Autoassociativa (*Autoencoder*) é produzir na saída uma representação que seja a mais próxima possível do dado inserido na entrada da rede. Um ponto que diferencia essa rede das outras já apresentadas é que o ajuste dos pesos da rede é feito de forma não supervisionada, uma vez que a qualidade dos pesos é verificada a partir da análise da reconstrução do dado comparado ao valor inserido anteriormente. Dividindo-a em uma estrutura codificador-decodificador, o processo de aprendizado tem como objetivo ajustar os pesos correspondentes à primeira fase, a fim de produzir uma representação codificada que possa ser recuperada na segunda fase, cujos pesos são ajus-

tados para viabilizar a reconstrução dos dados inseridos. Devido a essa estrutura, é desejável que uma rede autoassociativa consiga extrair alguma informação relevante dos dados inseridos, uma vez que uma cópia dos dados por toda a extensão da rede não seria útil para a resolução de problemas, embora essa situação satisfaça a condição exposta.



(a) Rede autoassociativa subdimensionada.

(b) Rede autoassociativa sobredimensionada.

Figura 4.9. Exemplos de redes autoassociativas. A rede é dita (a) subdimensionada quando a dimensão das camadas ocultas são menores que a dimensão do dado de entrada. (b) Caso a dimensão das camadas ocultas seja maior do que a do dado e entrada, a rede é dita sobredimensionada.

Usualmente uma rede autoassociativa é composta por camadas distribuídas simetricamente e respeitando o modelo codificador-decodificador [Charte et al., 2018]. Além da divisão quanto à profundidade, como presente nas demais redes (rasas e profundas), é possível caracterizar as redes autoassociativas quanto ao número de neurônios na camada intermediária. Quando a dimensão das camadas ocultas é menor que a do dado de entrada as redes são ditas Subdimensionadas (*Undercomplete*) [Goodfellow et al., 2016, Charte et al., 2018]. As redes autoassociativas subdimensionadas são comumente empregadas em problemas nos quais é desejável a redução da dimensionalidade do dado analisado, uma vez que a rede é obrigada a aprender representações mais compactas do dado inserido. Um exemplo de rede autoassociativa subdimensionada pode ser visto na Figura 4.9(a). Quando as camadas ocultas possuem dimensões maiores que a do dado inserido diz-se que a rede é Sobredimensionada (*Overcomplete*) [Goodfellow et al., 2016, Charte et al., 2018]. Essa estrutura permite a extração de mais características dos dados, produzindo uma representação em uma dimensão maior do que a do dado de entrada [Rifai et al., 2011]. A representação de uma rede autoassociativa sobredimensionada está ilustrada na Figura 4.9(b). Nessa situação, há uma preocupação adicional com o treinamento da rede, uma vez que existe uma possibilidade considerável de a rede aprender a replicar o dado de entrada por toda a extensão da rede. Para evitar esse problema é empregada uma variação da rede, chamada **Rede Autoassociativa Regularizada**, que inibe a tendência de cópias entre camadas adicionando um regularizador à função custo.

A rede **Autoassociativa Empilhada** (*Stacked Autoencoder*) é outra variação de rede neural que faz uso de redes autoassociativas. Nessa variação, constrói-se a rede neural a partir de camadas treinadas separadamente. Cada camada é tratada como uma rede autoassociativa separada, que são conectadas posteriormente, "empilhando-as" [Hinton e Salakhutdinov, 2006, Bengio et al., 2007]. Dessa forma, a implementação dessa rede consiste em um pré-treinamento não supervisionado das camadas separadamente seguido de um treinamento conjunto da rede como um todo em uma etapa de refinamento dos

parâmetros (*fine tuning*). Essas redes foram essenciais para popularizar as redes neurais profundas devido à facilidade de implementação. Elas evitam problemas associados ao treinamento de redes muito profundas, como alta variação nos parâmetros da rede, o que pode fazer com que resultados que não são ótimos sejam encontrados, além de evitar o sobreajuste. Atualmente, com o predomino do uso de aprendizado profundo em problemas de aprendizado supervisionado e com a descoberta de abordagens mais eficientes para o treinamento de redes profundas, como o *Dropout* [Srivastava, 2013], é observado um gradual abandono dessa variação de rede neural pela comunidade de aprendizado profundo [Goodfellow et al., 2016], ficando restrita a situações com pequenos conjuntos de dados [LeCun et al., 2015]. Exemplos de trabalhos recentes relacionados às Redes Desafiadoras [Lv et al., 2015, Zhang et al., 2018, Aceto et al., 2019a] ainda utilizam redes autoassociativas empilhadas. O sucesso dessas redes pode ser atribuído à presença de um grande número de amostras não classificadas nos problemas estudados. Em um trabalho para verificar se o pré-treinamento não supervisionado, como visto no treinamento das redes autoassociativas empilhadas, Paine et al. verificam que em problemas com muitas amostras não rotuladas, o ajuste da rede neural é melhorado, especialmente quando combinada a técnicas mais modernas [Paine et al., 2014]. LeCun et al. apontam para a possibilidade de um aumento de trabalhos que envolvam aprendizado não supervisionado em suas previsões para o futuro do aprendizado profundo [LeCun et al., 2015], o que pode fazer com que essa rede volte a ser mais utilizada.

Além da variante empilhada, uma outra variação de rede autoassociativa comumente utilizada é a **Autoassociativa Contrativa** (*Contractive Autoencoder*). Esse tipo de rede encoraja o aprendizado de representações mais robustas às variações nos dados de entrada. Considerando como se cada n características do dado analisado estivessem representadas em um dado espaço n, a rede busca favorecer as direções de maior variação, que são mais significativas no dado, enquanto aquelas que são menores ou mais raras são contraídas [Rifai et al., 2011]. Forçando que os pesos sejam determinados de tal forma que $W' = W^T$, onde W e W' são os pesos do codificador e decodificador, respectivamente, o aprendizado da rede consegue ter um maior controle em forçar as duas estruturas a serem contrativas [Alain e Bengio, 2014]. Outra rede amplamente utilizada, a **Autoassociativa Eliminadora de Ruído** (*Denoising Autoencoder*), tem como objetivo aprender a recuperar uma representação não corrompida do dado inserido [Goodfellow et al., 2016]. Considerando f e g respectivamente como as funções de codificação e decodificação, pode-se dizer que a rede autoassociativa eliminadora de ruído busca robustez na reconstrução do dado, isto é, tem como objetivo encontrar a melhor $(g \circ f)(x)$, enquanto a contrativa encoraja uma melhor representação $f(x)$ [Rifai et al., 2011]. Por fim, outra rede autoassociativa regularizada popular é a **Rede Autoassociativa Esparsa** (*Sparse Autoencoder*), que é tipicamente empregada para aprender características do dado analisado para desempenhar tarefas como classificação [Goodfellow et al., 2016], usando um regularizador que força a esparsidade na codificação gerada.

4.3. Aprendizado Profundo Aplicado a Redes Desafiadoras

As aplicações de aprendizado de máquina costumam seguir o fluxo de trabalho ilustrado na Figura 4.10. As etapas principais são descritas a seguir:

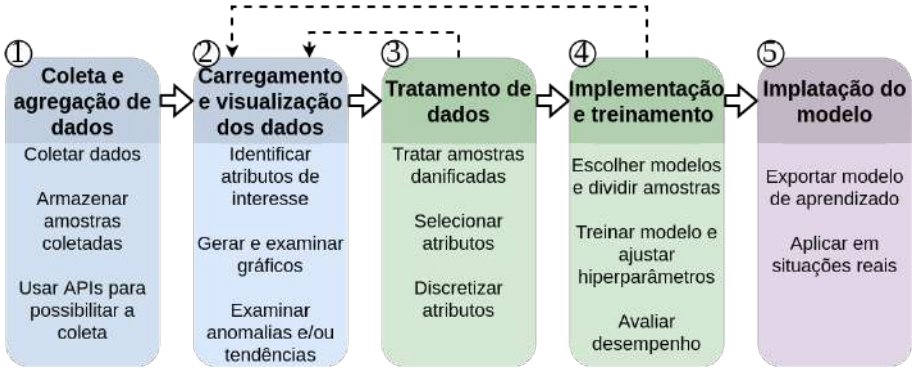


Figura 4.10. Fluxo de trabalho típico em aplicações utilizando aprendizado de máquina.

1. **Coleta e agregação de dados:** os dados são coletados a partir de diversas fontes, como sistemas IoT, formulários online, sítios *web* públicos, dentre outros.

2. **Carregamento e visualização dos dados:** ferramentas para geração de gráficos e análises estatísticas são usadas para analisar os dados em alto nível. Dessa forma, é possível identificar anomalias e inconsistências nos dados, além de padrões e tendências.

3. **Tratamento e pré-processamento dos dados:** com a identificação das anomalias e inconsistências, é possível tratar e pré-processar os dados que servirão de entrada para os algoritmos de aprendizado de máquina. Diversos algoritmos são usados nessa etapa e não é foco deste minicurso apresentá-los. O pré-processamento dos dados é crucial para alcançar um bom desempenho do modelo.

    - **Tratamento de amostras danificadas e formatação de atributos:** nesta etapa são tratados os dados faltantes decorrentes de um processo de coleta de dados imperfeito e valores em escalas diferentes são ajustados para uma escala única exigida por alguns algoritmos para obtenção de melhores resultados. Além disso, os atributos devem ser adequados aos algoritmos utilizados. Por exemplo, para algoritmos que recebem como entrada valores numéricos, os valores categóricos devem ser previamente convertidos. Também é necessário analisar a correlação dos atributos para possível redução em sua quantidade, ocasionando consequente melhora do tempo de treinamento e do resultado final do modelo. As transformações usadas devem ser aplicadas igualmente nos conjuntos de treino, validação e teste.
    - **Seleção de atributos com maior poder preditivo:** ferramentas estatísticas permitem identificar os atributos presentes nas amostras do conjunto de dados que são mais úteis para prever o alvo de cada amostra. Por exemplo, para a previsão da nota de um aluno é muito mais adequado utilizar a sua quantidade de horas de estudo do que a sua altura. Um algoritmo comum para composição e seleção de atributos é o de Análise de Componentes Principais (*Principal Component Analysis* – PCA), que transforma os atributos para que apenas os que possuírem maior capacidade de predição sejam selecionados.

- **Discretização (opcional):** o conjunto de dados pode ser discretizado para se adequar à entrada de um algoritmo de aprendizado de máquina. Essa discretização pode ser feita tanto de forma numérica quanto de forma categórica, dependendo do tipo de algoritmo a ser utilizado.

4. **Escolha e implementação do modelo de aprendizado:** a tarefa a ser desempenhada define o modelo ótimo para o cenário. A escolha do modelo deve considerar se o problema é supervisionado ou não supervisionado, se é um problema de classificação ou regressão, quais dispositivos são acessíveis para o treino e implantação do modelo etc.

    - **Início do treinamento:** o conjunto de dados é dividido nos conjuntos de treino, validação e teste. Após a escolha de hiperparâmetros, o modelo pode ser treinado e avaliado no conjunto de validação. Caso as métricas obtidas não sejam satisfatórias, os hiperparâmetros podem ser ajustados e uma nova rodada de treinamento pode ser executada. Apenas após a escolha final de hiperparâmetros, o modelo pode ser exportado e avaliado na etapa seguinte. A exportação do modelo significa que os pesos de uma rede neural ou os coeficientes de um regressor linear, por exemplo, estão definidos e o modelo pode ser usado em dados inéditos numa aplicação em tempo real.
    - **Avaliação dos resultados:** o conjunto de teste pode ser exposto ao modelo a fim de quantificar as métricas, como a acurácia em um problema de classificação, de forma apropriada. Ou seja, o modelo é aplicado em dados inéditos para garantir que o modelo obtém consistentemente o desempenho descrito.

5. **Implantação do modelo:** na última etapa, com o modelo treinado e com seu desempenho avaliado, é possível fazer a implantação do modelo em uma aplicação prática, seja para predição de carga, cálculo de probabilidade de falhas ou outras.

A seguir, este minicurso revisa trabalhos da literatura que aplicam nas Redes Desafiadoras o fluxo de trabalho descrito, de forma total ou parcial.

4.3.1. Redes Sem Fio Móveis

As comunicações através das redes sem fio vêm se popularizando, dado o crescente poder computacional dos dispositivos móveis e a consequente agregação de diferentes aplicações e serviços. Essa agregação, aliada à capacidade de mobilidade, acarreta a adesão cada vez maior dos usuários aos dispositivos móveis, provocando um aumento vertiginoso no volume de dados gerado em rede. Dessa forma, o aprendizado de máquina aplicado às **redes sem fio móveis** permite que tarefas desafiadoras como o gerenciamento dos recursos da rede, as análises em tempo real e o suporte ao crescente volume de dados sejam tratadas de forma a melhorar a experiência do usuário [Reis et al., 2020]. Dentre os trabalhos de aprendizado profundo aplicados às redes sem fio móveis, destacam-se o uso dos dados da infraestrutura ou do desempenho da rede para a previsão de informações da própria rede [Grando et al., 2019, Pierucci e Micheli, 2016] e para a classificação de tráfego [Aceto et al., 2019a, Nguyen e Armitage, 2008]. Além disso, as características da rede podem ser capturadas a partir da sua modelagem em forma de grafo, possibilitando

o estudo de seus componentes e das relações de comunicação. Essa modelagem permite determinar a importância de cada um dos nós para a operação da rede [Medeiros et al., 2017], ao identificar os nós que desempenham papéis estratégicos, como por exemplo, uma função central no controle da rede [Medeiros et al., 2016, Barbosa et al., 2019]. Esta seção discute pesquisas relevantes que aplicam o aprendizado profundo no cenário de redes sem fio móveis. O foco é dado a pesquisas relacionadas à análise de Qualidade de Experiência (*Quality of Experience* – QoE) e do tráfego de dados, que buscam aprimorar o gerenciamento ou a qualidade das redes sem fio móveis.

Predição de QoE do usuário

O aumento do uso dos serviços de Internet em ambientes sem fio faz com que os usuários busquem cada vez mais por serviços que ofereçam maior QoE. A resposta dos provedores é aprimorar a manutenção e operação da rede a fim de manter o nível da Qualidade de Serviço (*Quality of Service* – QoS) aceitável. Nessa direção, sabe-se que modelos de rede neural *Multilayer Perceptron* (MLP) são capazes de prever a QoE do usuário de redes móveis [Pierucci e Micheli, 2016] com base em Indicadores-Chave de Desempenho (*Key Performance Indicators* – KPIs). As estatísticas de QoS e QoE obtidas podem, por sua vez, ser usadas para guiar o desenvolvimento de novos mecanismos de controle e gerenciamento da rede utilizando, por exemplo, aprendizado por reforço [Bhattacharyya et al., 2019].

Tradicionalmente, a QoE do usuário é medida de forma subjetiva através de um grupo de voluntários participantes. No entanto, algumas características de QoS medidas a partir da operação em rede têm alta relação com a QoE do usuário e não são utilizadas como parâmetro adicional [Bhattacharyya et al., 2019]. A dificuldade de avaliar as características de QoS que podem melhorar a QoE se deve à incapacidade dos gerenciadores das redes de processar a grande quantidade de informação de QoS recebida pelo provedor de serviço. Dessa forma, o uso do aprendizado profundo se torna uma ferramenta capaz de avaliar diversas características de QoS para, então, estimar a QoE dos usuários. Nesse contexto, Pierucci e Micheli analisam uma base de dados de um provedor de serviço na Itália e destacam quais métricas de QoS medidas pelo provedor mais influenciam a QoE. A ideia é usar as métricas como entradas de um MLP a fim de melhorar a QoE [Pierucci e Micheli, 2016]. Inicialmente, divide-se o volume de dados e a vazão obtida pelos usuários em quatro regiões. Cada uma dessas regiões indica uma QoE diferente, que pode ser classificada de ruim até excelente. Os autores utilizam um MLP de duas camadas ocultas, uma com 4 neurônios e a outra com 7, e a saída da rede neural indica em qual das quatro regiões se encontra a QoE do usuário. A matriz de confusão mostra que o modelo proposto é capaz de classificar com alta precisão a QoE do usuário.

Em uma abordagem diferente, Bhattacharyya et al. focam no uso das estatísticas de QoE e QoS para melhorar o desempenho da transmissão de vídeo para os usuários. Para tanto, os autores desenvolvem a plataforma *QFlow*, baseada em aprendizado por reforço, que insere pacotes em diferentes filas de prioridade de um ponto de acesso (*Access Point* - AP). Os autores consideram um AP em situação de alta demanda trafegando dados de uma transmissão de vídeo do *YouTube*. O *YouTube* é considerado devido a sua grande quantidade de requisitos e por ocupar a maioria dos pacotes da Internet atualmente [Bhattacharyya et al., 2019]. Através de um agente controlador, a *QFlow* envia novas instru-

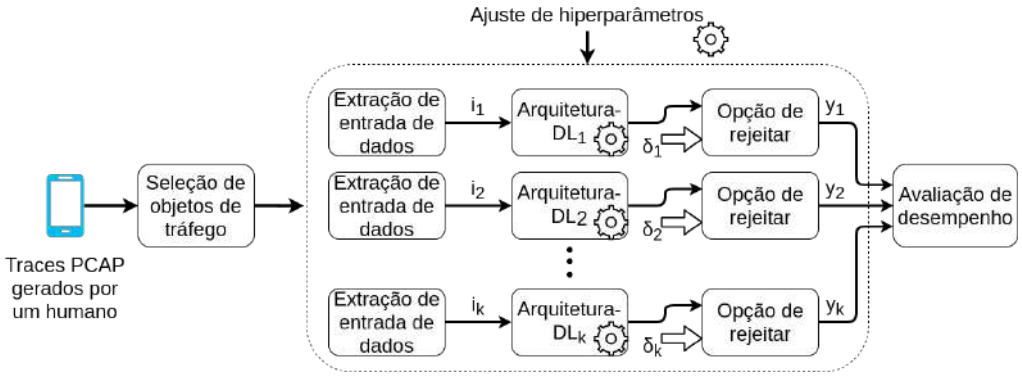


Figura 4.11. Arquitetura de ajuste e comparação de técnicas de aprendizado, adaptado de [Aceto et al., 2019a]. Os blocos de Extração de dados de entrada selecionam uma característica do tráfego e a transforma em dados de entrada para a arquitetura de aprendizado profundo. Essa arquitetura implementa um conjunto de técnicas distintas, avaliadas quanto ao desempenho em relação à predição de tráfego gerado por aplicativos de dispositivos móveis.

ções para atribuição de pacotes nas filas de prioridade do AP. As informações de QoE são obtidas a partir da aplicação do usuário, enquanto a QoS é obtida do AP. O agente de aprendizado por reforço usa a QoE como base para as recompensas e é constituído por uma rede neural MLP composta por duas camadas ocultas com 64 e 34 neurônios. Os autores mostram que a abordagem proposta alcança uma maior QoE quando comparada a outras abordagens de atribuições de filas como a *Vanilla* e *Round Robin*.

Análise do Tráfego da Rede

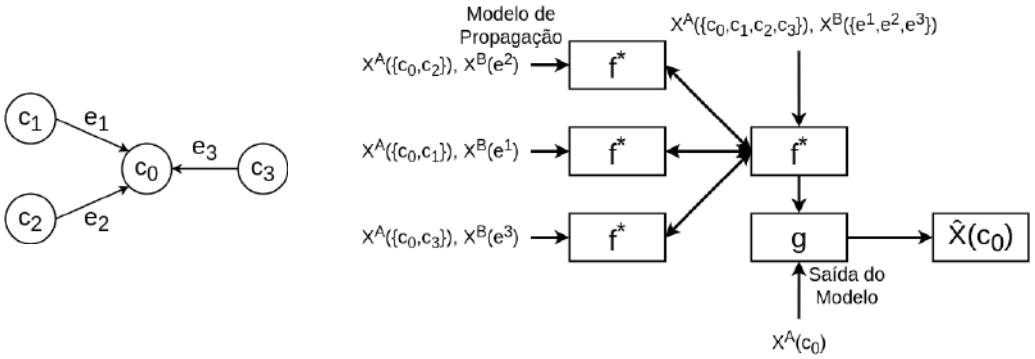
O rápido crescimento no tráfego das redes sem fio leva a estresses significativos do meio de comunicação. Com isso, há o surgimento de um grande desafio na alocação e no gerenciamento de recursos da rede, afetando diretamente a QoE dos usuários [Tang et al., 2017]. O uso do aprendizado profundo surge então como uma solução para analisar o grande volume de tráfego nas redes devido a sua capacidade de lidar com grandes quantidades de dados. Além disso, as aplicações usadas em dispositivos móveis sofrem constantes atualizações que alteram o conteúdo e a troca de dados, o que também desafia a realização tradicional da análise de tráfego. A predição do tráfego da rede pode auxiliar na alocação de recursos e, como consequência, garantir a QoS e o bom desempenho da rede [Wang et al., 2018]. Já a classificação do tráfego pode ser essencial para alguns serviços como detecção de intrusão, realocação de recursos ou a identificação dos recursos da rede utilizados pelos clientes [Nguyen e Armitage, 2008].

Aceto et al. avaliam como diferentes técnicas de aprendizado se comportam na classificação de tráfego com diferentes formas de entrada de dados. A ideia é possibilitar o uso futuro do aprendizado profundo na classificação de tráfego de dispositivos móveis [Aceto et al., 2019a]. A base de dados é um conjunto de tráfego de dados de usuários que utilizam diversas aplicações em sistemas *Android* e *iOS*. Dentre as aplicações estão o *Google Maps*, o *EFood*, o *Google Hangouts* e o *Crackle*. A Figura 4.11 ilustra a estrutura de ajuste e comparação das técnicas de aprendizado. Os dados de entrada são extraídos de diferentes formas e então aplicados a diversas técnicas de aprendizado. O desempenho

da avaliação é armazenado após analisar se a técnica é válida para os dados de entrada utilizados. A primeira e a segunda características extraídas dos dados são os primeiros bytes da carga útil do pacote e os primeiros bytes da carga total do pacote. Essas características são utilizadas como entrada para uma Rede Autoassociativa Empilhada (*Stacked Autoencoder* – SA) com cinco camadas, uma CNN com os dados de entrada em uma dimensão (1D-CNN), uma CNN com os dados de entrada em duas dimensões (2D-CNN) e uma *Long Short-Term Memory Network* (LSTM). A terceira característica extraída é baseada em campos do cabeçalho dos primeiros pacotes. Esses campos não contêm carga de dados criptografada e contêm informações como as portas de origem e destino e tamanho da janela TCP. Essas informações são usadas como dados de entrada para uma 2D-CNN, uma LSTM e um modelo híbrido no qual a saída de uma 2D-CNN é modelada como uma matriz e então utilizada como entrada de uma LSTM. Os resultados mostram que o desempenho da classificação de tráfego para o *Android* e o *iOS* é melhor quando se considera os primeiros bytes de carga útil dos pacotes e as arquiteturas que apresentam melhor desempenho são a 1D-CNN e a 2D-CNN. Vale ressaltar que todas as demais técnicas apresentam bons resultados, sendo assim, todas as técnicas são capazes de realizar a classificação do tráfego. Além disso, o estudo mostra que, apesar dos avanços, ainda é necessário maior aprofundamento para identificar uma arquitetura que sirva a todas as bases de dados com igual desempenho a partir de um modelo de entrada único. Nesse sentindo, as redes híbridas podem ser uma solução promissora.

A diversidade das aplicações e do comportamento dos usuários tornam a predição de tráfego em redes celulares desafiadora. Além disso, a mobilidade do usuário introduz uma dependência espacial, e o comportamento social, como a existência de um feriado, também influencia na demanda do tráfego da rede [Wang et al., 2018]. Apesar dessa dinamicidade e variação temporal, as Redes Neurais de Grafos (*Graph Neural Networks* – GNNs) são capazes de realizar a predição de tráfego nas redes celulares [Wang et al., 2018]. Na GNN, o conjunto de dados é modelado como um grafo para ser utilizado como entrada de uma rede neural profunda. Wang et al. usam uma GNN em um conjunto de dados composto de informações capturadas das torres da rede celular de uma cidade na China. Esse conjunto de dados possui informações importantes de cada transmissão de dados que ocorre através da rede celular, como a identificação do momento da criação do fluxo de dados, a identificação do usuário e da torre celular relacionada ao fluxo, a aplicação e o tipo de dispositivo utilizado pelo usuário. Os identificadores dos usuários são anonimizados para manter a privacidade. A informação de criação do fluxo dos dados captura a sua dependência temporal, enquanto a informação de identificação da torre captura a dependência espacial dos dados. O tráfego de dados entre os dispositivos móveis e a torre mais próxima é identificado como tráfego interno à torre (X^A), enquanto o tráfego entre as torres é identificado como tráfego externo à torre (X^B). Os dados de tráfego compõem um vetor de dados a cada período t que corresponde ao *downlink* de cada meia hora. A Figura 4.12(a) ilustra a representação em grafo da recepção de dados de uma torre (c_0). Cada uma das torres possui um tráfego gerado pelos dispositivos ligados diretamente a ela, as arestas (e_1, e_2 e e_3) representam os dados que estão sendo transmitidos pelas torres vizinhas à torre c_0 (c_1, c_2 e c_3). A Figura 4.12(b) é a representação do modelo de rede neural adotado. Nessa figura, os dados de entrada são X^A e X^B a cada instante t, a função f^* denota o modelo de propagação implementado por uma RNN de duas camadas ocul-

tas com 5 neurônios cada, e a função g é o modelo de saída implementado por um MLP composto por duas camadas ocultas com 6 neurônios cada.



(a) Grafo da recepção de dados de uma torre celular. Em C_i contém o trafégo X^A e em e_i contém o tráfego X^B.

(b) Arquitetura da técnica de aprendizado GNN. A função f^* denota uma RNN e g denota uma MLP.

Figura 4.12. Modelo de predição de tráfego para uma torre celular usando GNN, adaptado de [Wang et al., 2018]. (a) O tráfego da rede é modelado como um grafo, sendo composto de tráfego entre dispositivo móvel e torre (X^A), e entre torres (X^B). (b) O tráfego serve como entrada para a arquitetura da GNN, que é composta por uma RNN e um MLP.

Wang et al. treinam o modelo proposto e verificam seu desempenho com base no Erro Absoluto Médio (*Mean Absolute Error* – MAE). Os resultados mostram que a proposta apresenta melhor resultado quando comparada a outras redes neurais profundas, como uma LSTM. A verificação do impacto da dependência espacial é feita através da classificação das torres pela métrica de centralidade *Page Rank*, que considera as arestas ligadas à torre. As três primeiras posições mais bem classificadas pelo *Page Rank* são torres em locais de grande concentração de pessoas, como centros de compras e universidades. Por fim, os autores analisam o MAE ao longo de um dia para capturar a dependência temporal da abordagem proposta. Os resultados mostram que a proposta apresenta melhor desempenho comparada às demais redes utilizadas como comparação. O estudo realizado pelos autores mostra a importância de capturar as informações de espaço e tempo relacionadas ao tráfego para realizar a predição.

4.3.2. Redes IoT e de Sensores

A implementação de ambientes inteligentes que gerem maior conforto e sejam responsivos aos usuários em geral, sempre foi uma ambição da sociedade. A **Internet das Coisas** (*Internet of Things* – IoT) surge como a estrutura para alcançar essa ambição, buscando incorporar capacidade de comunicação aos dispositivos eletrônicos de forma a explorar o suporte da Internet para viabilizar a implementação dos ambientes inteligentes. Entende-se como uma rede IoT aquela formada entre dispositivos sensores e atuadores capazes de se comunicarem entre si e com a Internet [Gubbi et al., 2013]. Uma rede IoT, portanto, reutiliza conceitos tradicionais das **Redes de Sensores sem Fio** (*Wireless Sensor Networks* – WSNs) como base para o desenvolvimento de aplicações mais sofisticadas. As redes IoT empregam sensores de baixo custo econômico e energético para monitorar

um fenômeno de interesse. Dispositivos inteligentes com sensores embarcados como *smartphones* e *smartwatches* também compõem o ecossistema IoT, oferecendo aplicações personalizadas a partir da interação com servidores e outros dispositivos.

As redes IoT se caracterizam pela heterogeneidade dos dados gerados pelos dispositivos, uma vez que é habitual empregar sensores de diferentes tipos em um nó da rede. Um ponto crítico no processamento de dados em redes IoT é a qualidade dos dados gerados. Ademais, o tamanho e a disposição geográfica da rede implicam correlações espaciais e temporais entre as amostras que precisam ser levadas em consideração. Essas características podem ser exploradas utilizando técnicas de agregação de dados para, por exemplo, diminuir as transmissões na rede, aumentando consequentemente sua vida útil. Devido ao volume massivo de dados produzidos, o cenário IoT é um terreno fértil para o emprego das técnicas de aprendizado profundo. Contudo, é possível observar que o uso dessas técnicas em IoT ainda está em um estado inicial [Ma et al., 2019]. O restante desta seção apresenta algumas áreas de pesquisa relevantes para o cenário IoT. A localização passiva sem dispositivos e o reconhecimento de atividades humanas são as primeiras áreas abordadas, sendo relevantes para a implementação de sistemas inteligentes e aplicações diversas. Dessa forma, demonstra-se o uso de aprendizado profundo com o objetivo de melhorar o bem-estar do usuário. Ao final, são apresentados trabalhos que buscam melhorar a vida útil das redes, reduzindo e otimizando a quantidade de dados transmitidos.

Localização Passiva sem Dispositivo

A determinação da posição de pessoas e objetos em uma dada região é essencial para o desenvolvimento de muitas aplicações em sistemas IoT [Zafari et al., 2019]. Muitas das técnicas tradicionais, por exemplo aquelas que usam dados de GPS, necessitam que o objeto monitorado carregue algum dispositivo que viabilize a localização, além de geralmente exigir que o dispositivo monitorado tenha participação ativa no procedimento de localização [Youssef et al., 2007]. Como tal situação pode ser um inconveniente para algumas tarefas, uma alternativa é fazer a identificação de forma passiva, levando apenas em conta a interação do objeto com o ambiente.

Um paradigma com bastante relevância no cenário IoT, herdado das WSNs, é a Localização Passiva sem Dispositivo (*Device-Free Passive Localization* – DfP). Uma rede IoT pode ser disposta no ambiente de interesse para realizar tarefas de localização, explorando o fato de certas propriedades de sinais de radiofrequência serem alteradas em função de mudanças no ambiente [Youssef et al., 2007]. Portanto, implementar um sistema que identifique tais variações passa a ser uma alternativa interessante para localização passiva. Esses sistemas podem usar o aprendizado profundo com esse fim, sendo mais comum o uso de redes autoassociativas. Wang et al. desenvolve um sistema utilizando uma rede autoassociativa esparsa para identificar mudanças no ambiente monitorado para determinar não só a posição de indivíduos, mas também qual a atividade e gestos que são realizadas pelo mesmo [Wang et al., 2016a]. A Figura 4.13 mostra uma configuração típica de um ambiente para identificação passiva da localização de um objeto. Os sensores formam uma rede em que cada nó pode se comunicar com os outros. A partir da medição da Intensidade do Sinal Recebido (*Received Signal Strength* – RSS) entre cada nó, a rede autoassociativa é treinada para aprender a projetar essas medições em uma dimensão menor, extraindo as características principais dos dados. Após o treinamento, a porção de deco-

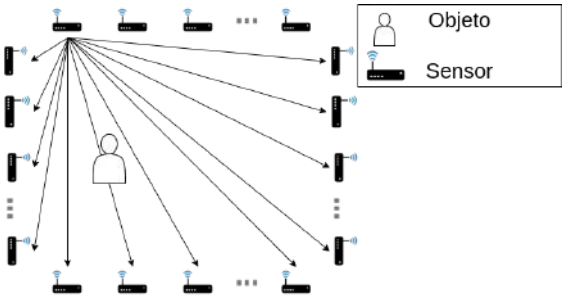


Figura 4.13. Exemplo de um sistema de localização passiva sem dispositivo que faz uso de RSS, adaptado de [Wang et al., 2016a, Zhao et al., 2019]. Os sensores medem a RSS entre ele e os demais sensores. A RSS entre os pares de sensores é modificada de acordo com a posição do objeto no ambiente e com a movimentação desse objeto, por exemplo uma pessoa executando uma atividade.

dificação da rede é descartada e a saída da nova última camada da rede autoassociativa é conectada a uma nova camada responsável por combinar os valores da saída da porção de codificação e executar a classificação. Com o acréscimo dessa camada, os parâmetros da rede são refinados, executando um novo treinamento. A abordagem baseada em aprendizado profundo difere de algumas técnicas mais tradicionais pela automação do ajuste dos parâmetros e viabilidade de identificar atividades e gestos simultaneamente, com menor sucesso na identificação de gestos.

Assim como o trabalho anterior, Zhao et al. exploram a intensidade do sinal recebido. No entanto, os autores usam uma abordagem que combina duas arquiteturas de redes neurais profundas para interpretar as variações de intensidade [Zhao et al., 2019]. Os autores usam uma rede neural autoassociativa convolucional para explorar as vantagens das redes convolucionais para o processamento de imagens e a capacidade da rede autoassociativa em ser treinada de forma não supervisionada. O sistema proposto coleta as variações percebidas por todos os sensores, armazenando os valores em uma matriz que caracteriza o estado de cada enlace entre nós da rede. Para isso, cada nó da rede transmite o sinal individualmente para os outros nós, que fazem as medições necessárias comparando-as com o valor do enlace não obstruído, isto é, o valor RSS sem um objeto.

A Figura 4.13 mostra os enlaces entre um único sensor e os demais. As medições são feitas para todos os sensores, formando uma matriz RSS. A matriz RSS obtida é convertida em uma imagem através do mapeamento de valores armazenados na matriz em *pixels*, permitindo explorar as características da parte convolucional da rede. Em seguida, a rede é pré-treinada com essas imagens, explorando a vantagem da estrutura autoassociativa da rede. É importante destacar que as camadas de decodificação da rede são compostas por camadas que desfazem as operações convolucionais e de *max pooling*, que são feitas nas camadas de codificação. Após o pré-treinamento, a parte correspondente ao decodificador é descartada e substituída por uma nova camada com a mesma função vista no trabalho anterior. Após o acréscimo dessa camada, uma nova fase de treinamento é executada para o refinamento dos parâmetros.

A rede convolucional autoassociativa é comparada às redes convolucional e autoassociativa que já são empregadas nesse tipo de problema. Para valores de Relação Sinal Ruído (*Signal-to-Noise Ratio* – SNR) variando entre -10 dB e 15 dB, a abordagem pro-

posta pelos autores apresenta maior acurácia na localização dos objetos em todos os cenários estudados. Aliado a isso, outro ponto destacado é o curto tempo de processamento da rede quando empregada no monitoramento, sendo necessários apenas 4 ms, tornando essa abordagem extremamente atrativa para localização *online*. Vale destacar, ainda, a necessidade de apenas dezesseis sensores para obter uma acurácia adequada quando a rede é empregada em um cenário com SNR igual a 0 dB.

Reconhecimento de Atividades Humanas

Muitas aplicações, sobretudo de interesse médico e de segurança, têm como objetivo identificar as atividades que são executadas por um indivíduo em um determinado espaço ou período. Essas atividades podem ser observadas através de sensores externos ao indíviduo, isto é, sensores que o indivíduo não carrega consigo [Wang et al., 2016a]; ou, alternativamente, através de sensores embutidos em dispositivos utilizados pelo indivíduo monitorado [Lara e Labrador, 2012], como aqueles pertencentes a *smartphones*, *smartwatches*, dentre outros. O Reconhecimento de Atividades Humanas (*Human Activity Recognition* – HAR) é uma área importante para o desenvolvimento de soluções de cuidados de saúde inteligentes (*Smart Healthcare*) [Ma et al., 2019], em que o exame de indivíduos é feito de maneira automatizada.

Hammerla et al. investigam o uso de diferentes redes neurais no reconhecimento de atividades a partir de dispositivos vestíveis ou dispositivos fixados nos usuários [Hammerla et al., 2016]. O objetivo é extrair informações acerca das vantagens e desvantagens de cada abordagem, uma vez que é raro um estudo mais detalhado de várias redes neurais com trabalhos que usam aprendizado profundo. Estes, normalmente, apresentam apenas o desempenho dos melhores sistemas. As redes são testadas em três bases de dados. A primeira consiste em medições de indivíduos executando atividades comuns em uma cozinha. A segunda possui registros de indivíduos executando atividades predeterminadas em uma ordem variada, sendo que o objetivo do aprendizado é classificar essas atividades. Por fim, a terceira base de dados é formada por medições de indivíduos portadores de Parkinson executando atividades que induzem um problema comum da doença, que consiste na dificuldade de iniciar um dado movimento. O objetivo geral do estudo é diferenciar determinadas situações nos três cenários. Os resultados obtidos mostram que as redes LSTM apresentam um desempenho superior às CNNs na identificação de atividades de curta duração, enquanto as CNNs apresentam desempenho superior em atividades de longa duração repetitivas, como correr e andar. As redes apresentam desempenhos variados de acordo com diferentes parâmetros, porém observa-se que redes sem realimentação apresentam a maior variação, o que exige uma exploração mais aprofundada de seus hiperparâmetros para produzir resultados satisfatórios, especialmente quando comparada às demais redes. Quando as técnicas comumente empregadas são comparadas, a rede do tipo LSTM bidirecional apresenta desempenho superior na base de dados com indivíduos em atividades normais em uma cozinha, com uma margem considerável. Porém, observa-se que o número de neurônios por camada afeta consideravelmente o desempenho da rede.

Similarmente a Hammerla et al., Ravi et al. focam na classificação de atividades. No entanto, os autores utilizam acelerômetros e giroscópios embutidos em um *smartphone* e têm como objetivo desenvolver um sistema complexo o suficiente para classificações mais qualificadas das atividades sem afetar demasiadamente os recursos

disponíveis no *smartphone* [Ravi et al., 2016a]. Em um trabalho anterior, Ravi et al. observaram que a complexidade da classificação é reduzida produzindo um espectrograma dos dados de entrada [Ravi et al., 2016b]. Algumas vantagens observadas do emprego dessa técnica são a invariância quanto ao tempo e à taxa de amostragem. Outra vantagem no emprego dessa técnica está no fato de atividades com alta variabilidade exibirem valores de espectrograma altos para varias frequências, enquanto atividades repetitivas apresentam valores altos em frequências específicas. O resultado do espectrograma serve de entrada para a rede neural. O modelo escolhido de rede neural é o sem realimentação, em que as conexões entre neurônios são limitadas de maneira similar à observada nas convolucionais. Outra decisão tomada é a de limitar o número de camadas escondidas, o que permite explorar construções hierárquicas do dado sem aumentar muito a complexidade do algoritmo. O treinamento é feito de maneira remota e a rede ajustada é carregada no celular do usuário para executar a classificação no próprio aparelho. Os resultados obtidos mostram uma acurácia superior ao modelo empregado no trabalho anterior, além de superar outros sistemas empregados para resolver o mesmo problema. Outro ponto relevante é o baixo tempo de computação encontrado, que viabiliza o reconhecimento de atividades humanas em tempo real.

Bianchi et al. focam no desenvolvimento de um sistema de monitoramento 24 h de pacientes. O objetivo é, dentro do paradigma de *Ambient Assisted Living* (AAL), identificar eventos anormais em idosos [Bianchi et al., 2019]. O sistema emprega sensores em dispositivos vestíveis que enviam os dados coletados a uma rede neural em um servidor para que as computações mais intensas sejam feitas remotamente. No trabalho, os autores comparam uma rede LSTM e uma CNN em três cenários. No primeiro cenário, as amostras são divididas de forma aleatória, respeitando 60% das amostras para o treinamento e o restante para teste. No segundo cenário, amostras de treino e teste pertencem a grupos de pessoas distintas, isto é, indivíduos presentes no conjunto de treinamento não estão no conjunto de teste. Por fim, no terceiro cenário, as amostras de todos os indivíduos estão presentes nos dois conjuntos. Constata-se que a CNN apresenta um desempenho superior ao da LSTM, principalmente no terceiro cenário. Dessa forma, os autores sugerem que para cada novo usuário, uma pequena fase de treinamento seja executada para aperfeiçoar os parâmetros da rede para o indivíduo. Comparando a rede convolucional com os métodos tradicionalmente utilizados, resultados compatíveis são observados, confirmando a viabilidade do modelo proposto.

Técnicas para Coleta de Dados e Gestão Eficiente de Recursos

As restrições de recursos dos dispositivos IoT impõem preocupações com o desperdício de energia na transmissão de dados. Logo, é fundamental reduzir o número de transmissões para expandir a vida útil dos sensores e da rede. As técnicas de agregação e de fusão exploram características inerentes dos dados para combiná-los e podem ser usadas para diminuir a quantidade de dados a serem transmitidos. Ademais, devido à grande quantidade e à variedade nos dados gerados em redes IoT, é comum combinar as áreas de agregação e fusão com a de mineração de dados. Assim, é possível extrair informações relevantes da tarefa executada para melhorar o desempenho do sistema.

Abu Alsheikh et al. investigam o desempenho de três redes neurais autoassociativas para comprimir medições em uma rede de sensores [Abu Alsheikh et al., 2016]. A

primeira rede neural é uma rede autoassociativa subdimensionada simples, capaz de obter uma representação mais compacta do dado inserido a partir apenas de sua estrutura. Uma variação de rede autoassociativa denominada pelos autores de Autoassociativa Redutora de Pesos (*Weight Decaying Autoencoder*) também é estudada. Nessa rede, um regularizador que penaliza soluções com pesos altos nas matrizes de codificação e decodificação é acrescentado à função custo. Por fim, a terceira rede neural se trata de uma rede autoassociativa esparsa. O trabalho analisa cenários com diferentes taxas de compressão, explorando as relações espaciais ou temporais presentes nos dados. A rede é treinada a partir de dados coletados previamente, enviadas a uma base central que é responsável por determinar as matrizes responsáveis pelas operações de compressão e descompressão dos dados. No cenário puramente espacial, verifica-se que as redes propostas apresentam um desempenho melhor que o desempenho das técnicas tradicionalmente empregadas, como análise de componentes principais (PCA), transformada discreta de cosseno (DCT) e transformada rápida de Fourier (FFT); especialmente quando baixas taxas de compressão são utilizadas. No caso temporal, novamente observa-se um desempenho superior a técnica tradicionalmente empregada na compressão nesse cenário, no caso *Lightweight Temporal Compression* (LTC). Comparando as três redes autoassociativas, verifica-se um melhor desempenho da rede autoassociativa tradicional na reconstrução do dado comprimido. Os autores atribuem essa observação à estrutura da rede em si, acrescentando que o uso dos reguladores nas outras duas redes degradam a capacidade de reconstrução dos dados.

Em outra abordagem utilizando redes autoassociativas para compressão, Yu et al. utilizam um Veículo Aéreo Não Tripulado (VANT) como unidade de processamento da rede [Yu et al., 2018]. A Figura 4.14 ilustra o sistema proposto, em que são utilizadas redes autoassociativas eliminadoras de ruído treinadas na nuvem a partir de amostras previamente coletadas. Para explorar as características espacias das amostras coletadas, os sensores são agrupados em conjuntos com o auxilio do algoritmo *K-Means*, sendo então determinada uma rede autoassociativa para cada agrupamento. As matrizes de peso calculadas para o codificador e decodificador são armazenadas no VANT e na nuvem, respectivamente. O VANT sobrevoa a área monitorada, coleta os dados sensoreados pelos conjuntos de nós e comprime os dados antes de enviá-los para a nuvem. Por fim, com os valores referentes aos decodificadores, a nuvem é capaz de reconstruir os dados coletados. Os resultados indicam que o sistema apresenta desempenho melhor que o método baseado em Sensoriamento Compressivo (*Compressive Sensing*) ao qual é comparado. Em especial, o melhor desempenho é mais evidente em taxas de amostragens menores.

Wang et al. adotam um uso diferente do aprendizado profundo. O intuito é eliminar possíveis erros de medição provenientes de desvios nos sensores através da fusão dos dados coletados utilizando uma rede neural convolucional [Wang et al., 2017]. Em sensores em operação em um longo período de tempo, é comum que pequenos desvios de medição se acumulem, o que pode fazer com que o sistema de medição não funcione corretamente. A rede neural convolucional tem como objetivo extrair relações temporais e espaciais do dado coletado para eliminar os desvios nas medições. A rede é estruturada de modo que sua camada mais externa seja responsável por projetar as amostras coletadas em um espaço de características (*feature space*), enquanto as demais camadas são responsáveis por fundir esses dados a fim de eliminar os desvios. Como o tamanho do campo de

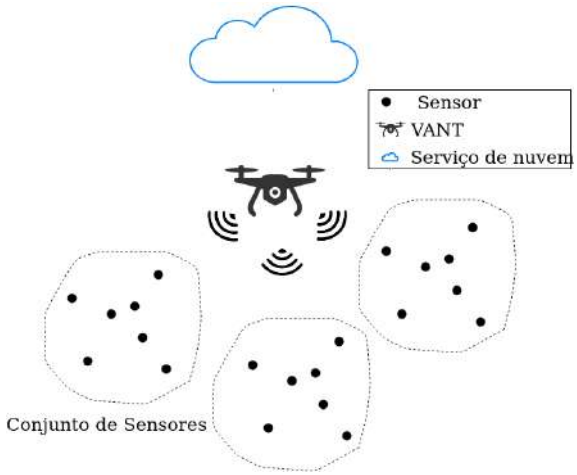


Figura 4.14. Sistema de agregação que utiliza VANT para coletar dados de uma área e comprimir os dados utilizando um serviço de rede autoassociativa eliminadora de ruído hospedado na nuvem, adaptado de [Yu et al., 2018].

recepção da camada convolucional é limitado, é desejável que dados relacionados estejam próximos na matriz de dados. Então, Wang et al. observam a necessidade de rearranjar as amostras coletadas de tal forma que os dados vindos de sensores próximos se encontrem em regiões adjacentes na matriz.A rede é treinada a partir de dados coletados previamente, de tal forma que os sensores são assumidos calibrados, isto é, contendo a capacidade de eliminar os desvios antes de entrarem em operação. Os autores adotam a estratégia de pré-treinar a rede inicialmente com dados que contenham pequenos desvios, seguido de um ajuste fino com dados com amostras que apresentam desvios maiores. Quando comparada a outros métodos de calibragem, a abordagem adotada apresenta uma maior taxa de recuperação, superando todos os cenários analisados. Além disso, no cenário em que menos de 15 sensores sofrem desvios, a proposta apresenta um erro menor quando a reconstrução é mal sucedida, embora apresente um erro superior nas reconstruções bem sucedidas quando comparado a uma técnica utilizada em um trabalho anterior [Wang et al., 2016b] que usa aprendizado Bayesiano. Wang et al. realizam uma análise adicional com diferentes modelos de ocorrência de desvios. A rede convolucional tem dificuldade em encontrar pequenos desvios, impedindo a obtenção do mesmo sucesso encontrado com desvios maiores. No entanto, a rede convolucional apresenta robustez a sobreajuste dos desvios devido à forma como é treinada. De modo geral, constata-se que o uso de redes convolucionais é adequado a problemas que envolvam longos períodos de monitoramento sujeitos a desvios nas medições.

4.3.3. Redes Industriais

A quarta revolução industrial, ou Indústria 4.0, consiste no atual processo de modernização da indústria, sendo baseado principalmente em sistemas ciberfísicos, análise de grandes volumes de dados (*big data*) e alta dependência da Internet das Coisas Industrial (*Industrial Internet of Things* – IIoT) e de tecnologias como computação em nuvem e em névoa [Aceto et al., 2019b]. O conceito de IIoT é um caso específico de IoT, que conecta os elementos industriais, como máquinas e sistemas de controle, com sistemas de informação e processos de negócios. Como consequência, um grande volume de dados é

coletado para que soluções analíticas sejam implementadas e operações industriais sejam otimizadas. Entre as principais diferenças entre IIoT e IoT estão o volume superior de dados no ambiente industrial e algumas diferenças em requisitos de comunicação como QoS, disponibilidade, segurança e privacidade. As tecnologias sem fio são tipicamente adotadas em IIoT para as aplicações industriais, as quais são capazes de alcançar grande flexibilidade e escalabilidade [Sisinni et al., 2018]. Assim, na Indústria 4.0, os sistemas de fabricação são capazes de monitorar os processos físicos e tomar decisões inteligentes através de comunicações em tempo real e interação com humanos, máquinas, sensores etc. Essa modernização permite que os processos de fabricação sejam reconfiguráveis, dinâmicos e flexíveis a fim de atender um mercado global e dinâmico [Zhong et al., 2017].

A Indústria 4.0 se relaciona a três conceitos-chave: produção inteligente, produção baseada em IoT e produção em nuvem [Zhong et al., 2017]. Na produção inteligente as decisões são baseadas em inteligência artificial e o processamento de grandes massas de dados (*big data*) é uma de suas tecnologias de suporte [Costa et al., 2012]. Na produção baseada em IoT, os dados provenientes de sensores precisam ser coletados em tempo real, assim como os processos de produção precisam ser visíveis e rastreáveis também em tempo real. Dessa forma, a tomada de decisão que afeta a produção é realizada com o menor atraso possível. Por fim, na produção em nuvem, os serviços podem ser distribuídos e compartilhados. Enquanto as informações são coletadas em toda fábrica, estas podem ser analisadas na nuvem e casos como defeitos na linha de produção ou temperaturas acima do padrão estabelecido podem ser detectados sem investir em equipamentos físicos ou utilizar poder computacional além do necessário para a execução da tarefa. Existem na literatura propostas que buscam aprimorar a produção utilizando redes neurais profundas. Por exemplo, a análise da confiança das aplicações sem fio na produção é desafiadora, dada a quantidade de dados a serem analisados. Ao mesmo tempo, é de grande importância, uma vez que as redes sem fio industriais são adotadas na IIoT. Nesse contexto, Sun e Willmann desenvolvem um sistema para avaliação de confiança [Sun e Willmann, 2019]. Já Zhang et. al propõem um sistema capaz de predizer a carga de trabalho na nuvem em intervalos futuros unindo a produção inteligente com o conceito de produção em nuvem [Zhang et al., 2018]. Li et al. propõem um sistema que une produção inteligente, produção baseada em IoT e produção em nuvem capaz de avaliar defeitos na produção com o auxílio de nós em névoa ou na nuvem a partir de dados coletados por diversas câmeras na linha de produção [Li et al., 2018].

Avaliação de Confiança em Redes Industriais Sem Fio

A confiança é um métrica que pode ser quantificada através de vários parâmetros. Sun e Willmann propõem um método para auxiliar a avaliação da confiança em aplicações sem fio industriais. A proposta foca na redução da dimensão dos parâmetros antes da fase de treinamento de um modelo de aprendizado. Dessa forma, tornam-se mais fáceis o processo de agrupamento das características das redes industriais e, consequentemente, a rotulação que auxilia a avaliação de confiança das aplicações [Sun e Willmann, 2019]. O modelo proposto pelos autores pode ser utilizado em diversos cenários envolvendo sistemas de comunicação sem fio, como sistemas de manutenção preditiva ou robôs industriais. Os dados de entrada são parâmetros de qualidade que podem ser mensurados, como o Tempo de Transmissão (*Transmission Time* – TT), ou calculados, como a Taxa

de Mensagens Perdidas (*Lost Message Ratio* – LMR), por exemplo. O funcionamento do sistema em um cenário genérico pode ser observado na Figura 4.15, onde os dados são inicializados, e também normalizados, antes mesmo do treinamento. A inicialização considera fatores de tempo, já que a confiança da rede muda dinamicamente. Esses dados são comprimidos pela rede profunda autoassociativa na fase de treinamento. Vale notar que após o treinamento, os dados passam apenas pela parte codificadora da rede autoassociativa antes que o agrupamento seja executado. O Algoritmo DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) é responsável por realizar o agrupamento. Os grupos são usados para calcular pontuações de confiança. O modelo é capaz de realizar o agrupamento e mostrar sua relevância uma vez que a quantidade de parâmetros para avaliar a confiança em aplicações sem fio industriais aumenta conforme o crescimento da complexidade das aplicações, tornando a avaliação da confiança mais difícil sem o auxílio de mecanismos como o proposto.

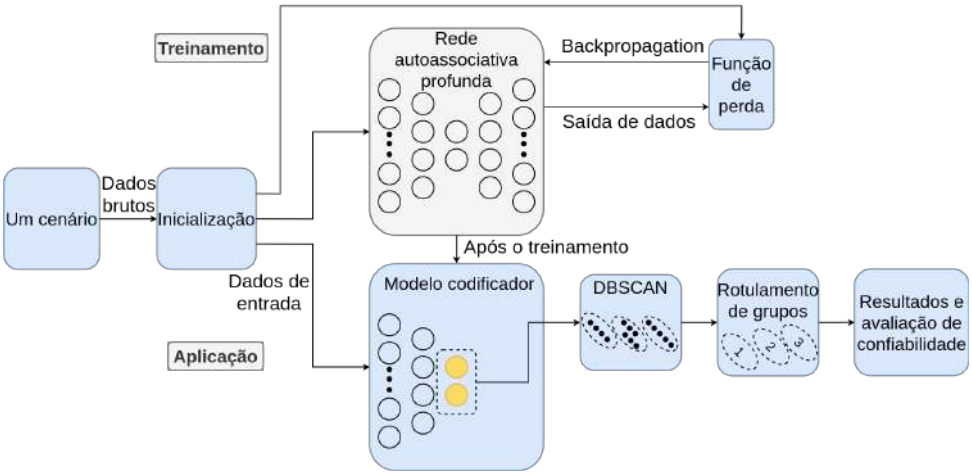


Figura 4.15. Processo para avaliação inteligente de confiança de aplicações sem fio, adaptado de [Sun e Willmann, 2019].

Predição da Carga de Trabalho na Nuvem

A computação em nuvem é uma das tecnologias empregada na modernização industrial por prover recursos computacionais e serviços sob demanda, além de possibilitar o armazenamento e processamento dos dados com baixo custo. Por ser uma das principais habilitadoras da Indústria 4.0, é importante avaliar e predizer a carga de trabalho na nuvem imposta pelas máquinas industriais. A predição da carga de trabalho nesse cenário é desafiadora, uma vez que as máquinas geram cargas de trabalho de forma dinâmica. A predição nesse caso permite que a qualidade dos serviços seja garantida e que os recursos da rede industrial sejam otimizados. Zhang et al. apontam que o treinamento de um modelo profundo é uma tarefa com alto consumo de tempo dado o grande número de parâmetros envolvidos e propõem um modelo para predição de carga de trabalho na nuvem adotando Decomposição Poliádica Canônica (*Canonical Polyadic Decomposition*) para comprimir os parâmetros [Zhang et al., 2018]. O modelo tem como objetivo predizer a utilização de CPU da máquina virtual com maior carga de trabalho em um dia futuro e a utilização de diversas máquinas virtuais em quatro intervalos futuros. A proposta busca também reduzir o tempo de execução e lidar com a grande quantidade de parâmetros.

A avaliação do treinamento é realizada através de quatro métricas: erro de aproximação, queda de acurácia da classificação, redução de parâmetros e aumento da velocidade de treinamento. O erro de aproximação e a queda de acurácia são causados pela conversão de parâmetros. A redução dos parâmetros é a taxa entre o número de parâmetros originais e comprimidos. Já o aumento de velocidade de treinamento refere-se à taxa entre o tempo de execução do treinamento tradicional das redes autoassociativas empilhadas com o do modelo proposto. O modelo proposto é comparado com o tradicional e com outro modelo que realiza a compressão de parâmetros através de outro método, conhecido como *Tucker decomposition*. Para avaliar a acurácia de predição, a comparação é feita com as técnicas tradicionais de redes neurais e de *Deep Belief Networks* (DBNs). O modelo proposto pelos autores alcança velocidade de treinamento superior aos demais com baixa perda de acurácia. A predição de utilização de CPU com maior carga de trabalho é realizada com maior acurácia que as outras técnicas de aprendizado de máquina.

Inspeção de Produtos através de Classificadores

Ao unir os conceitos de IIoT e aprendizado profundo, e utilizar a computação na nuvem, Li et al. desenvolvem um sistema capaz de identificar e classificar defeitos na linha de produção de forma inteligente, tendo a sua fabricação monitorada por diversas câmeras. Dessa forma, o sistema busca operar em tempo real, apesar da grande quantidade de dados gerada pelas câmeras. Para alcançar seus objetivos, os autores utilizam nós em névoa em conjunto com nós na nuvem para reduzir o tempo de execução [Li et al., 2018]. Os autores ressaltam que as soluções baseadas em redes neurais profundas melhoram a análise e reconhecimento de padrões para a detecção de defeitos na linha de produção, mas possuem o poder computacional como empecilho para implementação de um sistema em tempo real. Uma das possibilidades para solucionar esse problema é implementar as redes neurais profundas em servidores na nuvem, mas essa abordagem completamente *online* pode aumentar o tempo de execução. Dessa forma, o sistema proposto adota também a computação *offline* através de nós em névoa.

O sistema proposto por Li et al. identifica defeitos nos produtos e mensura o grau de gravidade, classificando-os como dentro ou não das conformidades de acordo com as políticas da fábrica. A Figura 4.16 mostra o funcionamento do modelo proposto. Inicialmente as imagens captadas por câmeras na linha de produção são enviadas aos computadores locais, ou seja, os nós da névoa. Em seguida, os dados passam por duas camadas de redes convolucionais que estão nos nós locais para tentar classificar os defeitos dos produtos. Os dados resultantes da classificação realizada pelos nós locais são chamados de resultados intermediários. Paralelamente, os resultados são enviados para os servidores na nuvem, passando por mais duas camadas convolucionais e duas camadas completamente conectadas, sendo possível realizar tanto a classificação dos defeitos quanto medir o grau de comprometimento através de um regressor. O resultado das camadas dos nós em névoa é considerado como final caso a abordagem *offline* seja capaz de analisar corretamente as imagens da câmera. Então os resultados intermediários deixam de ser enviados à nuvem. O sistema determina se a abordagem *offline* é ou não capaz de realizar a análise utilizando um limite definido como o máximo aceitável para a função de custo. Esse resultado final obtido sem os servidores na nuvem é chamado de *early exit*, uma vez que o sistema realiza a avaliação de defeitos mais rapidamente. É possível adotar mais de uma *early exit* na

abordagem *offline*, enviando os dados paralelamente não apenas aos servidores na nuvem, mas também para outras camadas convolucionais. Uma camada de *extra max pooling* é utilizada para reduzir o tamanho dos dados que são enviados aos servidores.

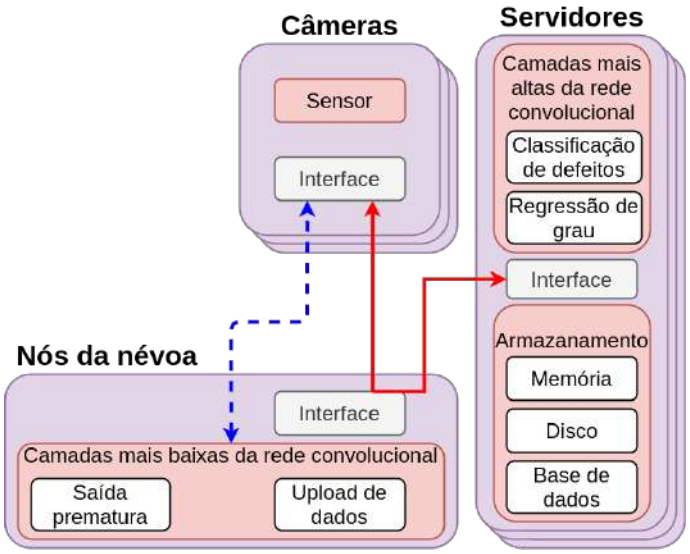


Figura 4.16. Sistema para avaliação de defeitos na linha de produção, adaptado de [Li et al., 2018].

Um dos desafios enfrentados pelos autores é o desenvolvimento de uma função de custo que permita o treinamento eficiente do regressor e do classificador simultaneamente. A função de custo *online* proposta é a soma de três outras funções e seus respectivos pesos com objetivos distintos: identificar o defeito através de uma função de custo baseada na função *softmax*, mensurar o grau de comprometimento do defeito e reduzir o sobreajuste. A função de custo da proposta *offline* é o somatório das funções de custo de cada saída prematura (*early exit*), sendo essas iguais à função de custo da proposta *online*. O conjunto de dados capturado manualmente para testar o sistema é composto por dez categorias de defeitos distintos. A habilidade de diagnóstico do sistema de classificação é comparada com dois outros métodos utilizados na detecção de defeitos, a detecção de contornos (*contour detection approach*) e a detecção baseada em pixels (*pixel-based method*), através da curva Característica de Operação do Receptor (*Receiver Operating Characteristic* – ROC). Em pelo menos nove situações o sistema dos autores possui melhor desempenho e reduz o tempo de execução em relação à computação local.

4.3.4. Redes Veiculares

As informações veiculares e do fluxo de veículos nas vias permitem aos sistemas de transporte inteligentes (*Intelligent Transportation Systems* – ITS) diversas análises para proporcionar melhor qualidade, conforto e segurança nas viagens. Nesse sentido, o aprendizado profundo pode ser adotado em redes veiculares para aprimorar o aprendizado da dinâmica da rede e a alocação e o gerenciamento de recursos. O aprendizado da dinâmica da rede pode envolver grandes quantidades de dados para que seja possível o desenvolvimento de sistemas inteligentes capazes de aprimorar a predição do fluxo de veículos ou a precisão da análise de segurança nas estradas [Lv et al., 2015, Peng et al., 2018].

Além disso, as redes neurais profundas são adotadas também para alocar recursos na comunicação entre veículos (*Vehicle-to-Vehicle* – V2V), para permitir que a potência para transmissão e sub-banda sejam otimizadas sem a necessidade de requisitar ou aguardar informações globais do sistema [Ye et al., 2019].

Aprendizado da Dinâmica da Rede

Os sistemas de transporte inteligentes viabilizam diversas análises sobre o tráfego veicular, que permitem definir a trajetória de uma viagem de forma a reduzir os congestionamentos e a emissão de gás carbônico [Lv et al., 2015]. Além disso, as informações sobre o tráfego veicular permitem avaliar e buscar formas de aumentar a segurança dos condutores e pedestres nas estradas [Peng et al., 2018]. Nesse contexto, Lv et al. propõem um modelo de aprendizado profundo utilizando redes autoassociativas empilhadas para que as características genéricas do fluxo de tráfego sejam aprendidas e o comportamento futuro possa ser predito [Lv et al., 2015]. Após a rede autoassociativa empilhada, a arquitetura emprega um preditor na camada de saída. O modelo proposto é realizado para fazer a predição do fluxo de tráfego em quatro intervalos futuros: 15 minutos, 30 minutos, 45 minutos e 60 minutos. Em cada um dos intervalos pretendidos é utilizado um número diferente de neurônios por camadas, determinado a partir de testes experimentais.

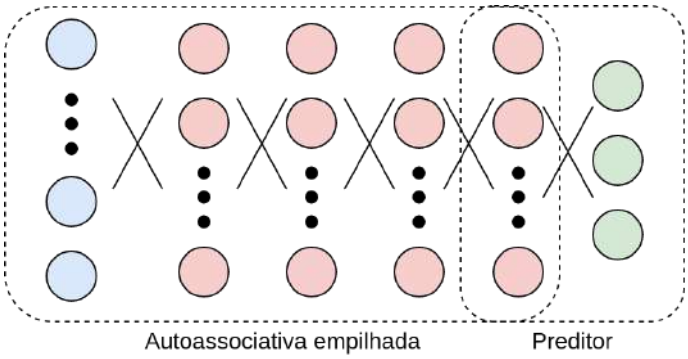


Figura 4.17. Arquitetura do modelo de predição do fluxo de tráfego, adaptado de [Lv et al., 2015].

A Figura 4.17 apresenta a arquitetura do modelo proposto pelos autores. Inicialmente, as características dos dados de entrada são extraídas pela rede autoassociativa empilhada antes de passar pelo preditor. Os dados que formam o conjunto de dados de entrada são coletados periodicamente por diversos detectores distribuídos pelos sistemas rodoviários da Califórnia. Antes de serem submetidos à rede autoassociativa empilhada, as informações obtidas por cada detector individualmente são agregadas em intervalos maiores. Em rodovias com múltiplos detectores, os dados de tráfego coletados pelos diferentes sensores são agregados para gerar o fluxo médio da rodovia. Na fase de treinamento do modelo, as camadas são treinadas das mais próximas da entrada para a saída conforme a Figura 4.17. O objetivo é minimizar a função de custo antes de treinar a próxima camada com a saída da camada anterior. A saída da última camada da rede autoassociativa empilhada é usada como entrada da camada de predição. O preditor inicializa seus parâmetros aleatoriamente ou através de treinamento supervisionado. Por fim, os parâmetros são ajustados em todas as camadas utilizando *backpropagation* de forma supervisionada.

Os autores avaliam o desempenho da proposta a partir de três indicadores: Erro Médio Absoluto (*Mean Absolute Error* – MAE), Erro Médio Relativo (*Mean Relative Error* – MRE) e Erro Quadrático Médio (*Mean Squared Error* – MSE). O modelo é analisado de forma comparativa com outras abordagens para predição de tráfego, como redes neurais com Função de Base Radial *(Radial Basis Function* – RBF). A proposta dos autores alcança acurácia média superior aos outros modelos comparados. O melhor desempenho obtido pela proposta dos autores é, em parte, resultado do ajuste adequado de hiperparâmetros, como o número de camadas escondidas e a quantidade de neurônios em cada camada. Dessa forma, o experimento comprova que o dimensionamento correto dos hiperparâmetros é importante para obter os melhores resultados possíveis.

Enquanto Lv et al. focam na definição de trajetórias, Peng et al. usam uma rede neural híbrida para aumentar a segurança de condutores e pedestres [Peng et al., 2018]. As pesquisas sobre a segurança veicular geralmente são feitas através de duas abordagens distintas. Na primeira, informações veiculares para análise de segurança dos condutores e pedestres são utilizadas; enquanto na segunda, imagens e fatores externos para analisar a segurança das estradas são consideradas. No entanto, as abordagens para análise da segurança nas estradas com base em fatores externos que utilizam modelos matemáticos podem não refletir bons resultados em todos os ambientes urbanos por assumir dados e parâmetros empíricos. A análise das imagens não possui grande precisão por focar apenas em imagens locais. Os autores tentam minimizar esses problemas propondo o *deep learning framework* (DeepRSI), uma rede neural híbrida que melhora a precisão da análise de segurança nas estradas, através da captura tanto das informações contidas nos sensores de carros e dispositivos, quanto de fatores externos do ambiente, como o clima [Peng et al., 2018]. O conjunto de dados de entrada para o modelo é um conjunto de dados urbanos da cidade de Nova Iorque, que inclui dados climáticos, eventos, e dados coletados por 13 mil táxis que apresentam as rotas por GPS e fornecem informações dos sensores dos veículos.

A Figura 4.18 ilustra o arcabouço DeepRSI composto por três redes convolucionais e uma rede totalmente conectada (*Fully Connected* – FCs) para cada região da cidade dividida em malha. O aprendizado espaço-temporal é atribuído às redes convolucionais. Para tanto, o conjunto de dados é dividido nos segmentos temporais "recentes", "próximos" e "distantes". Cada segmento é enviado a uma rede convolucional distinta, que realiza o aprendizado espacial a partir das camadas convolucionais densas. As saídas são então agregadas através de um método de fusão baseado em matriz paramétrica. A parte completamente conectada recebe como dados de entrada os fatores externos do ambiente e sua saída é adicionada às saídas convolucionais. A função de ativação ReLu (*Rectified Linear Unit*) tem como objetivo tornar a convergência mais rápida e, por fim, a função de ativação *softmax* faz a classificação para a saída. A saída é o índice de segurança (*Safety Index* – SI) de cada região. Assim, o SI permite avaliar a segurança das rodovias levando em consideração todos os aspectos importantes, ou seja, tanto as informações relacionadas aos veículos e pedestres, quanto as informações relacionadas ao ambiente.

A avaliação do DeepRSI é feita através da análise das métricas de **precisão** e **sensibilidade (*recall*)**. Os resultados mostram que o DeepRSI apresenta melhor desempenho quando comparado a outras abordagens como Árvore de Decisão (*Decision Tree* – DT) e Máquina de Vetor de Suporte (*Support Vector Machine* – SVM). Os autores defendem

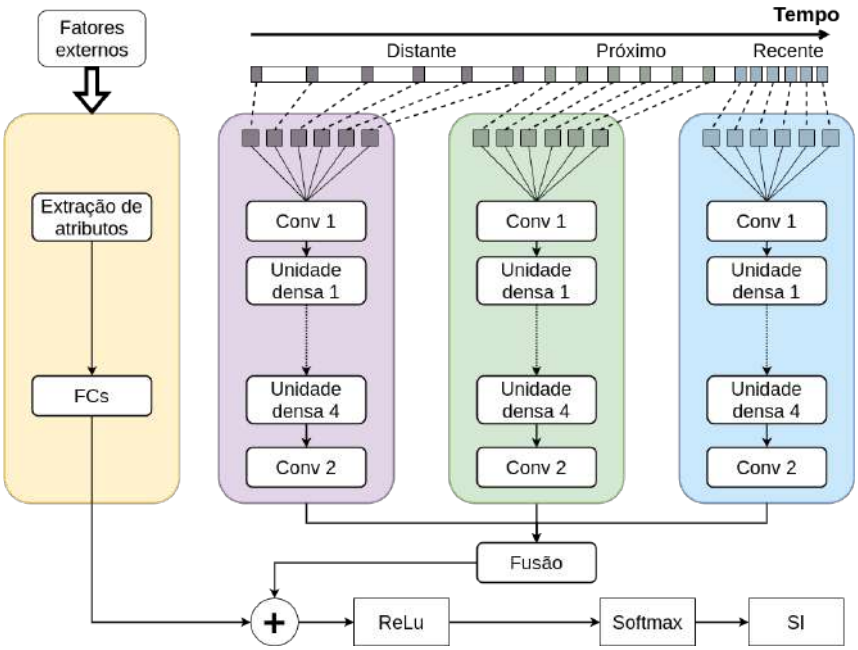


Figura 4.18. DeepRSI é um arcabouço que usa uma arquitetura híbrida para avaliar a segurança em estradas através do seu índice de segurança (SI), adaptado de [Peng et al., 2018]. As redes convolucionais recebem como entrada os dados relacionado ao veículos, e a rede completamente conectada recebe as informações relacionadas a fatores externos, como o clima.

que o arcabouço proposto é suficientemente genérico para ser aplicado em qualquer cenário, pois seu aprendizado não está limitado a imagens locais somente, como em outras abordagens relacionadas à segurança nas estradas. O fator mais importante do DeepRSI é a agregação dos dados geralmente utilizados para a segurança de condutores e pedestre para análise de segurança nas estradas, o que implica em maior refinamento da análise.

Alocação e Gerenciamento de Recursos

As comunicações nas redes veiculares podem ocorrer fundamentalmente entre veículos (V2V) ou entre veículo e infraestrutura (*Vehicle-to-Infrastructure* – V2I). Mais recentemente, porém, surgiu o paradigma V2X (*Vehicle-to-Everything*), no qual a comunicação ocorre entre veículos e outros sistemas de comunicação como as redes celulares. Independentemente do cenário, a comunicação veicular é essencial para reforçar a segurança nas estradas. No entanto, essa comunicação é complexa e sofre com diversas interferências inerentes ao ambiente de comunicação sem fio. A alocação de recursos é uma abordagem comumente utilizada em redes sem fio de uma forma geral para melhorar a comunicação entre os dispositivos. Nas redes veiculares, alguns autores propõem o uso de modelos de aprendizado profundo para promover uma alocação de recursos mais eficiente. Normalmente, os mecanismos de alocação de recursos para comunicação V2V são centralizados, recaindo em um problema de otimização NP-difícil. Além disso, a centralização do mecanismo pode levar a uma sobrecarga de controle na rede. Para evitar esses problemas, pode-se utilizar uma abordagem descentralizada, que tende a ser mais autônoma e robusta. Nesse contexto, Ye et al. propõem um sistema descentralizado baseado em aprendizado profundo por reforço para promover a comunicação eficiente em

uma VANET (*Vehicular Adhoc NETwork*) [Ye et al., 2019]. Os autores implementam um agente autônomo capaz de tomar decisões e determinar a sub-banda e a potência de transmissão que otimizam a comunicação no momento. Isso é feito sem que o agente requisite informações globais.

A proposta de Ye et al. usa uma função de recompensa que destaca o contraste entre a comunicação V2V e a V2I. O objetivo é garantir os requisitos de latência do V2V e, ao mesmo tempo, garantir que a comunicação V2V não interfira na comunicação V2I. O modelo proposto usa *Deep Q-Learning* para alocar os recursos na rede. A proposta pode ser aplicada em cenários *unicast* ou *broadcast*. No cenário *unicast*, as mensagens frequentemente não alcançam todos os veículos em uma vizinhança, já no cenário *broadcast* elas podem prejudicar a comunicação devido ao excesso de colisões de pacotes. Aplicando o modelo proposto ao cenário *unicast* a cada instante de tempo t, o agente observa o estado da comunicação e, de acordo com a política de aprendizado, realiza a ação de selecionar uma sub-banda e a potência da transmissão. Já no cenário *broadcast*, além das ações para a transmissão, o agente também retransmite uma mensagem que ele próprio recebeu anteriormente por *broadcast*, dentro de um subconjunto de mensagens existentes.

Os treinos e testes do modelo proposto são feitos para um ambiente simulado. Os veículos são dispostos na via aleatoriamente e supõe-se que cada veículo precisa estabelecer uma comunicação com outros três veículos. A quantidade de veículos varia de 20 a 160 durante o experimento. A arquitetura do modelo usa uma rede neural completamente conectada com cinco camadas, sendo três camadas ocultas com 500, 250 e 120 neurônios. O modelo proposto é comparado a outros métodos em ambos os cenários *unicast* e *broadcast*, e apresenta melhor aproveitamento dos enlaces de comunicação. No entanto, o modelo proposto apresenta elevado tempo de computação devido à complexidade, sendo necessário promover melhorias para reduzir esse tempo e viabilizar seu uso.

4.3.5. Segurança nas Redes Desafiadoras

A grande quantidade de dispositivos existentes nas Redes Desafiadoras com capacidade de comunicação e acesso à Internet aumenta significativamente a superfície de ataque da rede. Se não existirem mecanismos de proteção eficientes, os dispositivos nessas redes constituem uma porta de entrada para usuários maliciosos. Em redes industriais, por exemplo, a proteção de sistemas de controle conectados à Internet é primordial para evitar novas formas de ataque às estruturas críticas. Nas Redes Desafiadoras, os Sistemas de Detecção de Intrusão (*Intrusion Detection Systems* – IDSs) são uma necessidade de segurança básica comum a todos os paradigmas e são factíveis utilizando aprendizado profundo. A vantagem dos IDSs baseados em aprendizado profundo sobre aqueles que utilizam técnicas clássicas de aprendizado de máquina é a redução na taxa de falsos positivos e a menor dificuldade em identificar novos tipos de ataques [Al-Hawawreh et al., 2018]. Esta seção discute os pontos mais importantes no desenvolvimento de sistemas de segurança nas Redes Desafiadoras, porquanto muitas das decisões tomadas ao abordar um tipo de rede permeiam as outras redes discutidas.

Al-Hawawreh et al. propõem um sistema de detecção de anomalias cuja arquitetura é baseada em redes autoassociativas e redes neurais sem realimentação. Os conjuntos

de dados utilizados na avaliação da proposta são compostos por dados coletados a partir de tráfego TCP/IP, divididos em três subconjuntos. Enquanto o subconjunto A apenas possui amostras do comportamento normal da rede, os subconjuntos B e C possuem amostras normais e de ataques [Al-Hawawreh et al., 2018].

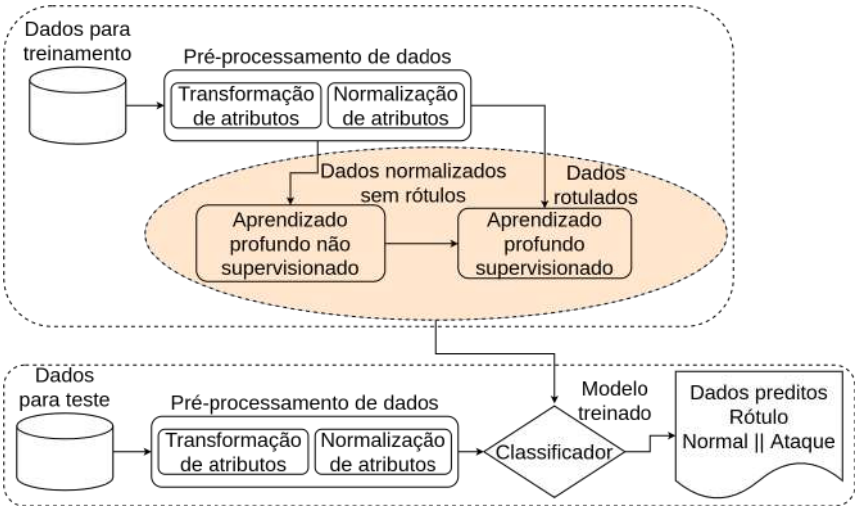


Figura 4.19. Arquitetura do sistema de detecção de anomalias em redes industriais, adaptado de [Al-Hawawreh et al., 2018].

A Figura 4.19 descreve o funcionamento do sistema proposto, onde inicialmente os dados são transformados e normalizados antes de iniciar a fase de treinamento e teste. Na transformação, os valores não numéricos são mapeados em valores numéricos e a normalização adotada é a *Z-score*, que mapeia os valores em torno da média do conjunto. O treinamento é realizado em duas fases. O objetivo da divisão do treinamento é fazer com que o algoritmo responsável pela classificação seja inicializado com pesos e limiares de ativação (*biases*) ótimos, ao invés de inicializar esses parâmetros com valores aleatórios. A aleatoriedade pode incorrer em mais tempo necessário para a convergência do algoritmo. A primeira etapa do treinamento ocorre com o auxílio de redes profundas autoassociativas utilizando o subconjunto de dados A. Na segunda fase, a acurácia do método é testada com as amostras do subconjunto B, que inclui dados de ataque. O erro médio quadrático é definido como a função de custo a ser minimizada na fase de treinamento, utilizando o gradiente descendente estocástico. Ao final dessa etapa o modelo para predição está pronto para ser testado. Por fim, o subconjunto C é utilizado na fase de testes. Os autores avaliam o sistema usando os conjuntos de dados NSL-KDD e UNSW-NB15. Os resultados mostram que a proposta alcança melhor desempenho no cenário NSL-KDD, obtendo 98.6% de acurácia, 99% de taxa de detecção e apenas 1.8% de taxa de falsos positivos. No UWSN-NB15, a proposta também apresenta bom desempenho, com acurácia de 92.4%, taxa de detecção de 93% e 8.2% de taxa de falsos positivos.

Diferentemente do trabalho anterior, Sharafaldin et al. investigam o desempenho de uma técnica de aprendizado profundo comparada a modelos tradicionais de aprendizado de máquina para a detecção de anomalias [Sharafaldin et al., 2018]. A abordagem dos autores implementa o modelo MLP. Os autores utilizam também um modelo de regressor de floresta aleatória (*random forest regressor*) para extrair o melhor conjunto de

atributos dentre os 78 atributos originais de um conjunto de dados. Dessa forma, é possível inferir quais atributos possuem maior **poder de predição** em relação ao alvo. Os atributos escolhidos variam de acordo com o tipo de ataque realizado na rede. É interessante ressaltar que a etapa de pré-processamento indica os atributos mais efetivos para a identificação e classificação de ataques, mesmo sem a aplicação do modelo profundo. Na segunda etapa, os autores utilizam seis modelos de aprendizado de máquina tradicional e o modelo MLP de aprendizado profundo. Os modelos são avaliados segundo as métricas de precisão, sensibilidade (*recall*) e **pontuação F1** (*F1-Score*). Os resultados mostram que o modelo de aprendizado profundo proposto não atinge um desempenho satisfatório.

Analogamente ao trabalho anterior, Panwar et al. também utilizam modelos tradicionais de aprendizado, evidenciando a necessidade de exploração de modelos profundos e seu desempenho para o desenvolvimento de sistemas de detecção de anomalias [Panwar e Shailesh, 2019]. No entanto, apesar dos trabalhos de Sharafaldin et al. e Panwar et al. mostrarem que os modelos de detecção de anomalia baseados em técnicas de aprendizado clássicas apresentam melhor desempenho quando comparadas aos modelos baseados em aprendizado profundo, Vinayakumar et al. apresentam uma análise detalhada que evidencia o potencial de redes neurais profundas para a classificação de ataques em diferentes tipos de redes de computadores [Vinayakumar et al., 2019]. Além de uma análise extensiva sobre os conjuntos de dados NSL-KDD, UNSW-NB15 e CICIDS, os autores também realizam uma análise sobre o ajuste de hiperparâmetros do modelo de aprendizado profundo. Os autores alcançam uma acurácia de $96,2\%$ para classificação multiclasse e $96,3\%$ de acurácia para classificação no conjunto de dados CICIDS com redes neurais profundas regularizadas de 3 camadas e 1 camada. O melhor desempenho alcançado com modelos de aprendizado clássico é obtido com o modelo de floresta aleatória (*random forest*), com $94,4\%$ de acurácia na classificação multiclasse e $94,0\%$ de acurácia na classificação binária no mesmo conjunto de dados.

4.4. Atividade Prática (*Hands-On*): Detecção de Ataques em Redes IP Utilizando Redes Neurais Profundas

O objetivo da atividade prática é demonstrar o uso das técnicas de aprendizado profundo discutidas neste minicurso para realizar tarefas de classificação em redes locais. O fluxo de trabalho seguido é o mesmo apresentado na Seção 4.3. Os desempenhos resultantes dos modelos aplicados também são comparados em relação a sistemas baseados em regras. O conjunto de dados escolhido para análise foi o *Intrusion Detection Evaluation Dataset* (CICIDS 2017) [Sharafaldin et al., 2018][1] por ser de fácil acesso e permitir que quase todas as etapas de um processo de análise de dados sejam executadas. Os passos necessários que não são viáveis para execução no contexto da atividade prática, como coleta de dados, são mencionados e discutidos. Uma breve descrição dos dados é feita nesta seção, porém Panigrahi e Borah apresentam uma discussão detalhada sob o ponto de vista da efetividade de sistemas de detecção de intrusão [Panigrahi e Borah, 2018]. Vale mencionar que Panwar et al. realizam uma análise aprofundada sobre o conjunto de dados CICIDS [Panwar e Shailesh, 2019], porém o objetivo desta atividade é permitir que

[1] Uma descrição detalhada e *links* para *download* podem ser encontrados em `https://www.unb.ca/cic/datasets/ids-2017.html`.

o leitor coloque em prática os conceitos apresentados nas seções anteriores ao invés de buscar uma exploração profunda do conjunto de dados aqui estudado.

Todos os programas utilizados e descritos nesta seção estão disponíveis em um repositório GitHub[2]. O repositório deve ser clonado para facilitar a execução da atividade prática. Para tal, os seguintes comandos devem ser executados para facilitar a atividade:

- Clonar o repositório:
  `git clone https://github.com/kaylani2/minicurso_ml_sbrc2020`
- Baixar o conjunto de dados:
  `wget http://205.174.165.80/CICDataset/CIC-IDS-2017/Dataset/MachineLearningCSV.zip`
- A partir da raiz do repositório, descomprimir o conjunto de dados:
  `unzip MachineLearningCSV.zip`
- A partir da raiz do repositório, instalar os requisitos:
  `pip3 install -r requirements.txt`

O Algoritmo 1 mostra uma visão de alto nível do treinamento da rede neural para o estudo de caso. Trata-se de um exemplo que considera aprendizado supervisionado.

Algoritmo 1: Algoritmo geral de treinamento supervisionado.

Entrada: Conjunto de amostras rotuladas.
Saída: Matriz de pesos e limiar de ativação.
`/* Inicialização dos conjuntos de dados e da matriz de pesos e limiar de ativação */`
1 Dividir aleatoriamente o conjunto de dados entre conjuntos de treino, validação e teste.
2 Dividir o conjunto de treino em lotes (*batches*).
3 Inicializar aleatoriamente a matriz de pesos e limiar de ativação.
`/* Treinamento do modelo */`
4 **para cada** *lote em lotes* **faça**
5 Aplicar regularização;
6 Calcular o valor da função custo no lote atual;
7 Calcular o vetor gradiente no lote atual;
8 **enquanto** $(custo > valor\ arbitrário) \vee (número\ de\ épocas < épocas\ escolhidas)$ **faça**
9 Atualizar os pesos na direção contrária ao vetor gradiente.
10 **fim**
11 **fim**
12 **retorna** matriz de pesos e limiar de ativação;

4.4.1. Apresentação das Bibliotecas de Programação

Inicialmente as bibliotecas e pacotes utilizados na atividade prática são apresentados para familiarizar o leitor com a utilização das estruturas de dados e algoritmos discutidos nas seções anteriores.

[2] `https://github.com/kaylani2/minicurso_ml_sbrc2020`

Pandas, Numpy e *Matplotlib*

O **Pandas**[3] é uma ferramenta de código aberto para análise e manipulação de dados construída sobre a linguagem Python. O Pandas utiliza duas estruturas de dados primárias: `Series` para dados unidimensionais e `Dataframe` para dados bidimensionais, sendo que os `DataFrames` são normalmente utilizados para manipular dados rotulados em sistemas de aprendizado. A biblioteca provê ainda facilidade para manipulação de dados faltantes, métodos comumente utilizados ao trabalhar com múltiplos conjuntos de dados (*join*, *merge* etc.) e ferramentas para converter, agrupar, analisar e modificar o formato dos dados. Normalmente os dados obtidos são carregados na forma de `DataFrames` para uma análise superficial e pré-processamento de dados para serem posteriormente convertidos para outra estrutura de dados apropriada para as bibliotecas que implementam os modelos de aprendizado. O **Numpy**[4] é um pacote largamente utilizado para computação científica. Ele possui as estruturas de dados manipuladas pelas bibliotecas específicas para o aprendizado de máquina e aprendizado profundo. A biblioteca usa vetores (*arrays*) multidimensionais como principal estrutura de dados, conhecida como `ndarray`, comumente tratada pelo seu *alias* `array` ou `numpy.array`. A biblioteca inclui métodos para a manipulação e inspeção dos objetos, além da forma tradicional de acesso a *arrays* através de índices inteiros. O **Matplotlib**[5] é uma biblioteca para exibição de gráficos em múltiplos formatos. Ela pode ser utilizada através de diversas interfaces, porém neste minicurso será manipulada através de programas em Python. O Matplotlib utiliza o elemento `Figure` para exibição de gráficos e figuras através de eixos, no qual a interface `matplotlib.pyplot` é normalmente usada para criação de gráficos simples e interativos. O Matplotlib é uma ferramenta poderosa para análise inicial de conjuntos de dados que facilita a leitura dos resultados de um modelo de aprendizado.

Scikit-learn, TensorFlow e *Keras*

O **Scikit-learn**[6] é uma biblioteca de código aberto para programação de algoritmos de aprendizado de máquina em Python. Ele suporta diversos modelos de aprendizado clássico, porém não implementa modelos de aprendizado profundo. Ela é normalmente utilizada para pré-processamento de dados, além da avaliação de modelos através de métodos para validação cruzada e ajuste de hiperparâmetros. O **TensorFlow**[7] é também uma plataforma de código aberto para implementação de técnicas de aprendizado de máquina. Diferentemente do Scikit-learn, ela é especialmente projetada para aprendizado profundo. A biblioteca utiliza uma estrutura de dados característica chamada `tensor`, que é composta por *arrays* multidimensionais. As funções necessárias para os algoritmos de aprendizado, como o cálculo de produtos matriciais para o *backpropagation*, são otimizadas internamente. O TensorFlow pode ser utilizado genericamente para expressar etapas computacionais arbitrárias através de grafos ou ***flows***, mas é normalmente utilizado para tarefas de aprendizado de máquina e, especificamente, aprendizado profundo. O **Ke-**

[3] `https://pandas.pydata.org/`
[4] `https://numpy.org/`
[5] `https://matplotlib.org/`
[6] `https://scikit-learn.org/`
[7] `https://www.tensorflow.org/`

ras[8] é uma biblioteca de código aberto escrita em Python que age como interface para o TensorFlow, sendo uma camada adicional de abstração construída sobre a plataforma TensorFlow. Ao invés de utilizar diretamente a plataforma TensorFlow, é possível selecionar facilmente hiperparâmetros dos modelos profundos, como o número de camadas em uma rede neural, além de diversas funções custo, funções de ativação e otimizadores.

4.4.2. Etapas para a Implementação de um Projeto com Aprendizado Profundo

Esta seção descreve de forma resumida as etapas para a implementação de um projeto prático que utilize as técnicas de aprendizado profundo discutidas neste minicurso.

Aquisição, Carga e Visualização dos Dados

A primeira etapa para a implementação de um projeto com aprendizado profundo é coletar e armazenar um grande volume de dados, porém a coleta de dados está fora do escopo deste minicurso. Logo, em posse do conjunto de dados CICIDS, especificamente o arquivo `Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv`, é necessário carregar e inspecionar os dados. O arquivo está em formato `CSV` (*Comma-separated values*) e pode ser carregado para inspeção com a biblioteca Pandas. A busca pela visualização gráfica da correlação entre cada um dos atributos e o alvo é uma tarefa impraticável, uma vez que cada amostra contém 78 atributos. Porém, é possível utilizar métodos do pacote Pandas para obter análises estatísticas sobre o conjunto de dados, como o desvio padrão ou a média de um atributo.

Tratamento do Conjunto de Dados

É importante notar que o conjunto de dados possui 225.745 amostras rotuladas e menos de $0,01\%$ das amostras possuem informações faltantes. Algumas estratégias podem ser utilizadas para tratar amostras defeituosas como: substituir atributos faltantes pelos valores médios, medianos ou mais frequentes; e eliminar amostras sem rótulos. A fim de apresentar a utilização de bibliotecas para tratamento de dados, um modelo `SimpleImputer` da biblioteca Keras foi utilizada para percorrer o conjunto de dados e substituir os valores de atributos das amostras defeituosas pelo valor médio dos atributos, porquanto substituir esses valores por valores aleatórios pode alterar drasticamente o desempenho do modelo. Visto que alterar os rótulos de amostras desconhecidas *a priori* pode resultar em falsas conclusões sobre os dados examinados e que o número de amostras com atributos faltantes é baixo em relação ao número total de amostras, as entradas "defeituosas" do conjunto de dados também podem ser removidas antes do treinamento. Como o percentual de amostras defeituosas no conjunto de dados é muito pequeno em relação ao total de amostras, ambas as abordagens produzem o mesmo resultado.

Configuração da Rede Neural

A biblioteca Keras é utilizada como API para instanciar o modelo profundo. Uma arquitetura com múltiplas camadas ocultas e volume de neurônios progressivamente menor é escolhida. A primeira rede neural escolhida, parcialmente descrita no Código 4.1, é um perceptron de múltiplas camadas (MLP) com uma configuração de camadas densas e

[8] `https://keras.io/`

camadas *dropout*, responsáveis pela regularização durante o treinamento. As camadas de *dropout* escolhem aleatoriamente um número de neurônios em cada camada, de acordo com a probabilidade passada como parâmetro, para ser "desativado" durante uma etapa do treinamento, isto é, sempre produzirá saída nula. Isso tem como objetivo evitar que neurônios da rede sejam sobreutilizados para a tarefa de predição, de forma que o modelo não fique dependente de um conjunto fixo de neurônios.

Código 4.1. Nova instância de um *Multilayer Perceptron.*

```
################################################################
## Instanciar o modelo de aprendizado
################################################################
from keras.models import Sequential
from keras.layers import Dense, Dropout
numberOfClasses = 2 # Classificacao binaria: Benign, DoS
batchSize = 128
numberOfEpochs = 20
model = Sequential ()
model.add (Dense (512, activation = 'relu', input_shape= (78, )))
model.add (Dropout (0.2))
model.add (Dense (256, activation = 'relu'))
model.add (Dropout (0.2))
model.add (Dense (128, activation = 'relu'))
model.add (Dropout (0.2))
model.add (Dense (numberOfClasses, activation = 'softmax'))
print ('Model summary:')
model.summary ()
```

Treinamento e Teste do Aprendizado

O Código 4.2 apresenta o trecho necessário para configurar a rede e iniciar o treinamento. Notam-se as escolhas de função de perda, otimizador, taxa de aprendizado e métricas utilizadas durante o treinamento. É importante também notar outros hiperparâmetros utilizados durante o treinamento: tamanho dos conjuntos de treino, número de épocas de treinamento e proporção do conjunto de validação.

Código 4.2. Compilação e treinamento da Rede Neural.

```
################################################################
## Compilar a rede
################################################################
from keras.optimizers import Adam
model.compile (loss = 'sparse_categorical_crossentropy',
               optimizer = Adam (lr=0.001),
               metrics= ['accuracy'])

################################################################
## Treinar (fit) a rede
################################################################
history = model.fit (X_train, y_train,
                     batch_size = batchSize,
                     epochs = numberOfEpochs,
                     verbose = 1,
                     validation_data = (X_test, y_test))
```

Análise dos Resultados

A configuração escolhida, após o ajuste de hiperparâmetros, atinge uma acurácia superior a 90%. Os hiperparâmetros podem ser variados a fim de entender como a variação de arquitetura da rede neural afeta o desempenho do classificador. É importante notar que ajustes no número de neurônios por camada, número de camadas, funções de ativação e otimizadores afetam o desempenho do modelo em conjunto. Dessa forma, não é possível variar o número de neurônios em uma camada, por exemplo, para depois variar o número de camadas em busca do melhor desempenho.

Também é importante mencionar que o algoritmo de otimização escolhido (*Stochastic Gradient Descent* – SGD) possui natureza estocástica, isto é, existem múltiplos caminhos que o algoritmo pode seguir durante a otimização e o resultado pode variar, mesmo que minimamente.

4.5. Considerações Finais, Perspectivas e Problemas em Aberto

Este minicurso apresentou conceitos básicos de aprendizado de máquina essenciais para a compreensão do aprendizado profundo. O minicurso ressaltou que o aprendizado profundo é um caso especial de aprendizado de máquina, no qual o modelo de aprendizado é dividido em múltiplas camadas de processamento que se completam de forma hierárquica para resolver um problema complexo. Apesar de existirem outros modelos de aprendizado profundo, o foco deste minicurso foi nas redes neurais profundas, uma vez que essas redes são um componente fundamental desse tipo de aprendizado. Ademais, este minicurso apresentou uma atividade prática na qual foi possível observar os desafios e os benefícios do uso de algoritmos de aprendizado profundo para resolução de problemas complexos. Dentre os principais benefícios, destaca-se a capacidade de tratar um grande volume de dados e a possibilidade de ajustar os hiperparâmetros de um modelo para que ele opere de forma otimizada.

O aprendizado profundo já é amplamente aplicado no processamento de imagens e vem sendo cada vez mais aplicado nas chamadas Redes Desafiadoras. Essas redes são definidas neste minicurso como redes caracterizadas pela geração de um grande volume de dados e pela inerente complexidade de operação. Dentro dessa classificação estão as redes sem fio, IoT, veiculares e industriais. Além de discutir diversas pesquisas que usam o aprendizado profundo para resolver problemas característicos dessas redes, este minicurso também discutiu trabalhos que aplicam aprendizado profundo na resolução de problemas de segurança em redes, que são comuns a todas as Redes Desafiadoras. O minicurso apresentou a forma com que as abordagens utilizam o aprendizado profundo e como ocorre a extração de dados dos conjuntos de dados utilizados, destacando os desafios encontrados nos trabalhos. Um dos desafios comum a todas as redes, por exemplo, é a manutenção de uma base de dados atualizada que preserve a privacidade dos usuários e mantenha as informações importantes anonimizadas, enquanto garante, ao mesmo tempo, que a base de dados contenha todas as informações necessárias para o funcionamento adequado dos algoritmos de aprendizado profundo. Além disso, as informações não podem ser tendenciosas.

As propostas baseadas em redes neurais profundas são capazes de lidar com o crescimento exacerbado do volume de dados a serem analisados e com o aumento da

velocidade de execução dos modelos. A aplicação das diversas arquiteturas baseadas em redes neurais profundas também possibilita que as métricas de avaliação de desempenho, como a acurácia, sejam melhores quando comparadas às obtidas em modelos tradicionais de aprendizado de máquina. Esse melhor desempenho das arquiteturas de aprendizado profundo está relacionado ao ajuste fino dos hiperparâmetros.

Apesar dos resultados promissores alcançados nos trabalhos listados, ainda é necessário aprofundar a pesquisa sobre aprendizado profundo aplicado às Redes Desafiadoras em diversos aspectos. Por exemplo, uma das promessas do aprendizado profundo é permitir a utilização da base de dados sem que haja tratamento prévio. No entanto, isso ainda não é realidade com os algoritmos atuais, uma vez que os dados de entrada das redes neurais profundas ainda são previamente analisados. Além disso, o bom desempenho das soluções baseadas em aprendizado profundo depende do ajuste adequado dos hiperparâmetros para o problema investigado. Esse processo de ajuste é, muitas vezes, automatizado [Aceto et al., 2019a, Wu et al., 2019, Neary, 2018]. No entanto, a maioria dos estudos atuais utilizam métodos empíricos que podem não chegar a soluções ótimas. O uso de valores não ótimos para os hiperparâmetros em geral resulta em resultados ruins quando comparados aos resultados alcançados quando algoritmos de aprendizado clássico são utilizados. Ressalta-se que não existe um modelo único que atenda a todas as questões em aberto, portanto, é necessário ponderar sobre quais modelos devem ser utilizados para atender melhor o problema investigado. Essa investigação é complexa e requer a combinação de diversos tipos de redes neurais profundas para obter bons resultados. Dessa forma, o uso de algoritmos de aprendizado profundo nas Redes Desafiadoras ainda apresenta desafios de pesquisa a serem explorados.

Agradecimentos

Os autores gostariam primeiramente de agradecer os professores Heraldo Luís Silveira de Almeida e José Gabriel Rodríguez Carneiro Gomes por toda atenção gentilmente dispensada. Em seguida, os autores agradecem igualmente o apoio do CNPq; da Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ); da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES), Código de Financiamento 001; e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processos n^o 15/24494-8 e 15/24490-2.

Referências

[Abu Alsheikh et al., 2016] Abu Alsheikh et al. (2016). Rate-distortion balanced data compression for wireless sensor networks. *IEEE Sensors Journal*, 16(12):5072–5083.

[Aceto et al., 2019a] Aceto, G., Ciuonzo, D., Montieri, A. e Pescapé, A. (2019a). Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management*, 16(2):445–458.

[Aceto et al., 2019b] Aceto, G., Persico, V. e Pescapé, A. (2019b). A survey on information and communication technologies for industry 4.0: State-of-the-art, taxonomies,

perspectives, and challenges. *IEEE Communications Surveys & Tutorials*, 21(4):3467–3501.

[Al-Hawawreh et al., 2018] Al-Hawawreh, M., Moustafa, N. e Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of Information Security and Applications*, 41:1–11.

[Alain e Bengio, 2014] Alain, G. e Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593.

[Barbosa et al., 2019] Barbosa et al. (2019). Centralidade de proximidade por múltiplos caminhos disjuntos: Aplicação em redes de longa distância. Em *Anais do SBRC 2019*, volume 37, p. 88–101.

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D. e Larochelle, H. (2007). Greedy layer-wise training of deep networks. Em *Advances in neural information processing systems*, p. 153–160.

[Bengio et al., 1994] Bengio, Y., Simard, P. e Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Bhattacharyya et al., 2019] Bhattacharyya, R., Bura, A., Rengarajan, D., Rumuly, M., Shakkottai, S., Kalathil, D., Mok, R. K. e Dhamdhere, A. (2019). Qflow: A reinforcement learning approach to high qoe video streaming over wireless networks. Em *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, p. 251–260.

[Bianchi et al., 2019] Bianchi, V., Bassoli, M., Lombardo, G., Fornacciari, P., Mordonini, M. e De Munari, I. (2019). IoT wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment. *IEEE Internet of Things Journal*, 6(5):8553–8562.

[Charte et al., 2018] Charte et al. (2018). A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion*, 44:78–96.

[Comarela et al., 2019] Comarela et al. (2019). Introdução à ciência de dados: Uma visão pragmática utilizando python, aplicações e oportunidades em redes de computadores. Em *Minicursos do SBRC 2019*, chapter 6, p. 246–295. SBC.

[Costa et al., 2012] Costa et al. (2012). Grandes massas de dados na nuvem: Desafios e técnicas para inovação. Em *Minicursos do SBRC 2012*, chapter 1, p. 1–58. SBC.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

[Deisenroth et al., 2019] Deisenroth, M. P., Faisal, A. A. e Ong, C. S. (2019). *Mathematics for machine learning*. Cambridge University Press Cambridge.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y. e Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[Grando et al., 2019] Grando et al. (2019). Machine learning in network centrality measures: Tutorial and outlook. *ACM Computing Surveys*, 51(5):102:1–102:32.

[Graves, 2012a] Graves, A. (2012a). Long short-term memory. Em *Supervised sequence labelling with recurrent neural networks*, p. 34–42. Springer.

[Graves, 2012b] Graves, A. (2012b). Neural networks. Em *Supervised sequence labelling with recurrent neural networks*, p. 13–33. Springer.

[Gubbi et al., 2013] Gubbi, J., Buyya, R., Marusic, S. e Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660.

[Hammer, 2000] Hammer, B. (2000). On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1-4):107–123.

[Hammerla et al., 2016] Hammerla, N. Y., Halloran, S. e Plötz, T. (2016). Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*.

[Haykin, 1994] Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[Hinton e Salakhutdinov, 2006] Hinton, G. E. e Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., White, H. et al. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L. e Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

[Khan et al., 2018] Khan, S., Rahmani, H., Shah, S. A. A. e Bennamoun, M. (2018). A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1):1–207.

[Lara e Labrador, 2012] Lara, O. D. e Labrador, M. A. (2012). A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209.

[LeCun et al., 1995] LeCun, Y., Bengio, Y. et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[LeCun et al., 2015] LeCun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Li et al., 2018] Li et al. (2018). Deep Learning for Smart Industry: Efficient Manufacture Inspection System with Fog Computing. *IEEE Transactions on Industrial Informatics*, 14(10):4665–4673.

[Liang e Srikant, 2016] Liang, S. e Srikant, R. (2016). Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*.

[Lv et al., 2015] Lv et al. (2015). Traffic Flow Prediction with Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873.

[Ma et al., 2019] Ma, X., Yao, T., Hu, M., Dong, Y., Liu, W., Wang, F. e Liu, J. (2019). A survey on deep learning empowered IoT applications. *IEEE Access*, 7:181721–181732.

[Mao et al., 2018] Mao et al. (2018). Deep learning for intelligent wireless networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 20(4):2595–2621.

[Medeiros et al., 2016] Medeiros, D. S. V., Campista, M. E. M., Mitton, N., Dias de Amorim, M. e Pujolle, G. (2016). Weighted betweenness for multipath networks. Em *Proc. of the Global Information Infrastructure and Networking Symposium (GIIS '16)*, p. 1–6.

[Medeiros et al., 2017] Medeiros et al. (2017). The power of quasi-shortest paths: ρ-geodesic betweenness centrality. *IEEE Transactions on Network Science and Engineering*, 4(3):187–200.

[Medeiros et al., 2019] Medeiros et al. (2019). Análise de dados em redes sem fio de grande porte: Processamento em fluxo em tempo real, tendências e desafios. Em *Minicursos do SBRC 2019*, chapter 4, p. 142–195. SBC.

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill.

[Neary, 2018] Neary, P. (2018). Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning. Em *2018 IEEE International Conference on Cognitive Computing (ICCC)*, p. 73–77.

[Nguyen e Armitage, 2008] Nguyen, T. T. e Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, 10(4):56–76.

[Osherson et al., 1991] Osherson et al. (1991). A universal inductive inference machine. *Journal of Symbolic Logic*, 56(2):661–672.

[Paine et al., 2014] Paine, T. L., Khorrami, P., Han, W. e Huang, T. S. (2014). An analysis of unsupervised pre-training in light of recent advances. *arXiv preprint arXiv:1412.6597*.

[Panigrahi e Borah, 2018] Panigrahi, R. e Borah, S. (2018). A detailed analysis of cicids2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7:479–482.

[Panwar e Shailesh, 2019] Panwar, Lokesh, P. e Shailesh (2019). Implementation of machine learning algorithms on cicids-2017 dataset for intrusion detection using weka. *International Journal of Recent Technology and Engineering (IJRTE)*, 8:2195–2207.

[Peng et al., 2018] Peng, Z., Gao, S., Li, Z., Xiao, B. e Qian, Y. (2018). Vehicle safety improvement through deep learning and mobile sensing. *IEEE network*, 32(4):28–33.

[Pierucci e Micheli, 2016] Pierucci, L. e Micheli, D. (2016). A neural network for quality of experience estimation in mobile communications. *IEEE MultiMedia*, 23(4):42–49.

[Ravi et al., 2016a] Ravi, D., Wong, C., Lo, B. e Yang, G.-Z. (2016a). A deep learning approach to on-node sensor data analytics for mobile or wearable devices. *IEEE journal of biomedical and health informatics*, 21(1):56–64.

[Ravi et al., 2016b] Ravi, D., Wong, C., Lo, B. e Yang, G.-Z. (2016b). Deep learning for human activity recognition: A resource efficient implementation on low-power devices. Em *2016 IEEE 13th international conference on wearable and implantable body sensor networks (BSN)*, p. 71–76. IEEE.

[Reis et al., 2020] Reis, L. H. A., Magalhães, L. C. S., de Medeiros, D. S. V. e Mattos, D. M. F. (2020). An unsupervised approach to infer quality of service for large-scale wireless networking. *Journal of Network and Systems Management*. In Press.

[Rifai et al., 2011] Rifai, S., Vincent, P., Muller, X., Glorot, X. e Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. Em *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, p. 833–840.

[Sharafaldin et al., 2018] Sharafaldin et al. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. Em *International Conference on Information Systems Security and Privacy (ICISSP)*, p. 108–116.

[Sharma et al., 2019] Sharma, A., Vans, E., Shigemizu, D., Boroevich, K. e Tsunoda, T. (2019). Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific Reports*, 9.

[Sisinni et al., 2018] Sisinni, E., Saifullah, A., Han, S., Jennehag, U. e Gidlund, M. (2018). Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734.

[Srivastava, 2013] Srivastava, N. (2013). Improving neural networks with dropout. *University of Toronto*, 182(566):7.

[Sun e Willmann, 2019] Sun, D. e Willmann, S. (2019). Deep learning-based dependability assessment method for industrial wireless network. *IFAC-PapersOnLine*, 52(24):219–224.

[Tang et al., 2017] Tang, F., Mao, B., Fadlullah, Z. M., Kato, N., Akashi, O., Inoue, T. e Mizutani, K. (2017). On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control. *IEEE Wireless Communications*, 25(1):154–160.

[Vinayakumar et al., 2019] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A. e Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, 7:41525–41550.

[Wang et al., 2016a] Wang, J., Zhang, X., Gao, Q., Yue, H. e Wang, H. (2016a). Device-free wireless localization and activity recognition: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 66(7):6258–6267.

[Wang et al., 2018] Wang, X., Zhou, Z., Xiao, F., Xing, K., Yang, Z., Liu, Y. e Peng, C. (2018). Spatio-temporal analysis and prediction of cellular traffic in metropolis. *IEEE Transactions on Mobile Computing*, 18(9):2190–2202.

[Wang et al., 2017] Wang, Y., Yang, A., Chen, X., Wang, P., Wang, Y. e Yang, H. (2017). A deep learning approach for blind drift calibration of sensor networks. *IEEE Sensors Journal*, 17(13):4158–4171.

[Wang et al., 2016b] Wang, Y., Yang, A., Li, Z., Chen, X., Wang, P. e Yang, H. (2016b). Blind drift calibration of sensor networks using sparse bayesian learning. *IEEE Sensors Journal*, 16(16):6249–6260.

[Wu et al., 2019] Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H. e Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, 17(1):26 – 40.

[Ye et al., 2019] Ye, H., Li, G. Y. e Juang, B.-H. F. (2019). Deep reinforcement learning based resource allocation for v2v communications. *IEEE Transactions on Vehicular Technology*, 68(4):3163–3173.

[Youssef et al., 2007] Youssef, M., Mah, M. e Agrawala, A. (2007). Challenges: device-free passive localization for wireless environments. Em *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, p. 222–229.

[Yu et al., 2018] Yu, T., Wang, X. e Shami, A. (2018). Uav-enabled spatial data sampling in large-scale IoT systems using denoising autoencoder neural network. *IEEE Internet of Things Journal*, 6(2):1856–1865.

[Zafari et al., 2019] Zafari, F., Gkelias, A. e Leung, K. K. (2019). A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*, 21(3):2568–2599.

[Zhang et al., 2018] Zhang et al. (2018). An efficient deep learning model to predict cloud workload for industry informatics. *IEEE Transactions on Industrial Informatics*, 14(7):3170–3178.

[Zhao et al., 2019] Zhao, L., Huang, H., Li, X., Ding, S., Zhao, H. e Han, Z. (2019). An accurate and robust approach of device-free localization with convolutional autoencoder. *IEEE Internet of Things Journal*, 6(3):5825–5840.

[Zhong et al., 2017] Zhong, R. Y., Xu, X., Klotz, E. e Newman, S. T. (2017). Intelligent manufacturing in the context of industry 4.0: a review. *Engineering*, 3(5):616–630.