TrueNorth Ecosystem for Brain-Inspired Computing: Scalable Systems, Software, and Applications

Jun Sawada*, Filipp Akopyan*, Andrew S. Cassidy*, Brian Taba*, Michael V. Debole*, Pallab Datta*,
Rodrigo Alvarez-Icaza*, Arnon Amir*, John V. Arthur*, Alexander Andreopoulos*,
Rathinakumar Appuswamy*, Heinz Baier*, Davis Barch*, David J. Berg*, Carmelo di Nolfo*,
Steven K. Esser*, Myron Flickner*, Thomas A. Horvath*, Bryan L. Jackson*, Jeff Kusnitz*,
Scott Lekuch*, Michael Mastro*, Timothy Melano*, Paul A. Merolla*, Steven E. Millman*, Tapan K. Nayak*,
Norm Pass*, Hartmut E. Penner*, William P. Risk*, Kai Schleupen*, Benjamin Shaw*, Hayley Wu*,
Brian Giera†, Adam T. Moody†, Nathan Mundhenk†,
Brian C. Van Essen†, Eric X. Wang†, David P. Widemann†,
Qing Wu‡, William E. Murphy‡,
Jamie K. Infantolino§, James A. Ross§, Dale R. Shires§, Manuel M. Vindiola§, Raju Namburu§,
and Dharmendra S. Modha*

*IBM Research, †Lawrence Livermore National Laboratory,
‡U.S. Air Force Research Laboratory, §U.S. Army Research Laboratory

Abstract

This paper describes the hardware and software ecosystem encompassing the brain-inspired TrueNorth processor – a 70mW reconfigurable silicon chip with 1 million neurons, 256 million synapses, and 4096 parallel and distributed neural cores. For systems, we present a *scale-out* system loosely coupling 16 single-chip boards and a *scale-up* system tightly integrating 16 chips in a 4×4 configuration by exploiting TrueNorth's native tiling. For software, we present an end-to-end ecosystem consisting of a simulator, a programming language, an integrated programming environment, a library of algorithms and applications, firmware, tools for deep learning, a teaching curriculum, and cloud enablement. For the scale-up systems we summarize our approach to physical placement of neural network, to reduce intra- and inter-chip network traffic. The ecosystem is in use at over 30 universities and government/corporate labs. Our platform is a substrate for a spectrum of applications from mobile and embedded computing to cloud and supercomputers.

TrueNorth Ecosystem for Brain-Inspired Computing: Scalable Systems, Software, and Applications

Jun Sawada*, Filipp Akopyan*, Andrew S. Cassidy*, Brian Taba*, Michael V. Debole*, Pallab Datta*, Rodrigo Alvarez-Icaza*, Arnon Amir*, John V. Arthur*, Alexander Andreopoulos*, Rathinakumar Appuswamy*, Heinz Baier*, Davis Barch*, David J. Berg*, Carmelo di Nolfo*, Steven K. Esser*, Myron Flickner*, Thomas A. Horvath*, Bryan L. Jackson*, Jeff Kusnitz*, Scott Lekuch*, Michael Mastro*, Timothy Melano*, Paul A. Merolla*, Steven E. Millman*, Tapan K. Nayak*, Norm Pass*, Hartmut E. Penner*, William P. Risk*, Kai Schleupen*, Benjamin Shaw*, Hayley Wu*, Brian Giera†, Adam T. Moody†, Nathan Mundhenk†, Brian C. Van Essen†, Eric X. Wang†, David P. Widemann†, Qing Wu‡, William E. Murphy‡, Jamie K. Infantolino§, James A. Ross§, Dale R. Shires§, Manuel M. Vindiola§, Raju Namburu§, and Dharmendra S. Modha*

*IBM Research, †Lawrence Livermore National Laboratory, ‡U.S. Air Force Research Laboratory, §U.S. Army Research Laboratory

***Abstract*—This paper describes the hardware and software ecosystem encompassing the brain-inspired TrueNorth processor – a 70mW reconfigurable silicon chip with 1 million neurons, 256 million synapses, and 4096 parallel and distributed neural cores. For systems, we present a *scale-out* system loosely coupling 16 single-chip boards and a *scale-up* system tightly integrating 16 chips in a 4×4 configuration by exploiting TrueNorth's native tiling. For software, we present an end-to-end ecosystem consisting of a simulator, a programming language, an integrated programming environment, a library of algorithms and applications, firmware, tools for deep learning, a teaching curriculum, and cloud enablement. For the scale-up systems we summarize our approach to physical placement of neural network, to reduce intra- and inter-chip network traffic. The ecosystem is in use at over 30 universities and government/corporate labs. Our platform is a substrate for a spectrum of applications from mobile and embedded computing to cloud and supercomputers.**

I. Introduction

Neural networks are currently in the midst of a resurgence, fueled by advances in deep learning [1] that include scalable tools for training these networks [2], [3], [4], as well as novel algorithms that achieve high performance on a wide range of tasks, including visual object classification [5], speech recognition [6], language modeling [7], and natural language understanding [8]. On the other hand, neurosynaptic architectures, such as TrueNorth [9], have demonstrated orders of magnitude improvement in computational energy-efficiency and throughput [10]. More recently, it has been shown that deep learning algorithms can be modified to train neural networks for specialized architectures with low-precision neurons and synapses and still yield comparable classification accuracy [11], [12], [13]. This enables a new class of energy-efficient neural network applications, ranging from mobile and embedded systems to cloud and supercomputer platforms.

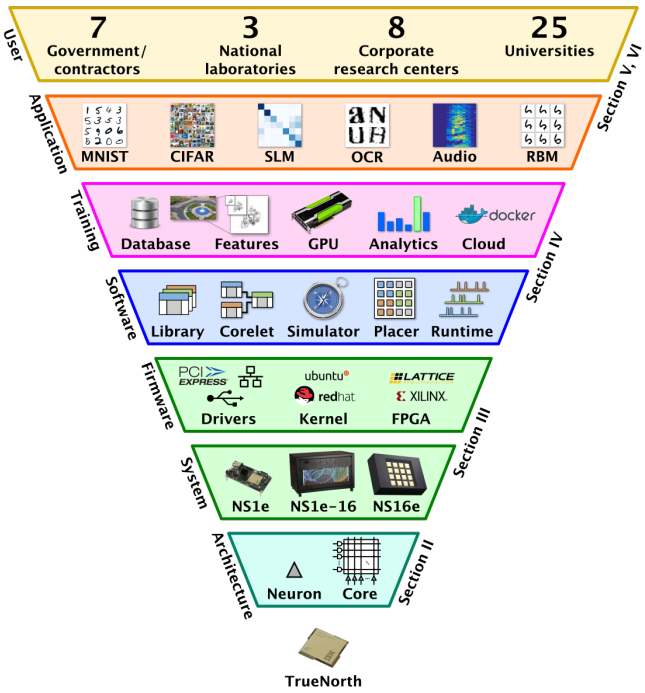


Fig. 1. Integrated, end-to-end, hardware/software ecosystem for large-scale neurosynaptic computing. The paper is organized describing the ecosystem layers in detail beginning at the bottom of the figure.

SC16; Salt Lake City, Utah, USA; November 2016
978-1-4673-8815-3/16/$31.00 ©2016 IEEE

At this juncture, what is missing from the neural network computational paradigm is a comprehensive ecosystem that realizes the full potential of this new architecture. In this work, we present an integrated, end-to-end novel ecosystem for specifying, training, and deploying neural networks based on the TrueNorth architecture, as shown in Fig. 1. This includes multiple hardware platforms and systems, seamlessly integrated with software languages, tools, libraries, environments, and offline training algorithms, all of which enable users to develop and deploy a wide range of applications.

The major novel technical contributions of this paper are:

- To demonstrate two new systems (Section III): the NS1e-16 *scale-out* system that consists of a network of 16 single-chip NS1e boards; and the NS16e *scale-up* system that consists of 4×4 array of TrueNorth chips.
- To describe the full development workflow including neural network training algorithms, libraries, a simulator, data preprocessors, model compilation, and other software tools. (Section IV).
- To provide a simple and effective algorithm for mapping logical network representations to physical hardware locations, in order to reduce intra- and inter-chip traffic (Section IV).
- And finally to provide deployment status of our system for over 100 developers at seven government agencies, three national labs, eight corporate research centers, and twenty-five universities. We include three sample applications (scale-out and scale-up) from our partners (Section V).

These systems–when programmed with convolutional networks [13]–achieve near state-of-the-art accuracy while delivering $>$ 6000 frames/sec/Watt, which is unmatched by any existing CPU, GPU, or FPGA platform (Section VII). Without an effective development flow, the Neovision application presented in [10] took $\sim$ 10 person-years of work to create. This paper presents the end-to-end ecosystem that allows engineers to achieve better results in person-weeks, and even person-days of effort. Thus, this paper represents a significant productivity improvement for the specialized hardware for neural networks. We have brought together chips, systems, firmware, development environments, and applications–all of the pieces required to realize future scaled systems based on brain-inspired computing.

II. TrueNorth

Synthesizing insights from neuroscience [14] and supercomputer simulations on large-scale cortical networks [15], [16], [17], [18], the TrueNorth chip was developed based on a parallel, event-driven, non-von Neumann kernel for neural networks that is efficient with respect to computation, memory, and communication [10]. The architecture is parallel, distributed, event-driven, modular, scalable, fault-tolerant, and co-localizes memory and computation to enable scalable, efficient, and flexible platforms for a wide variety of neural network applications [9].

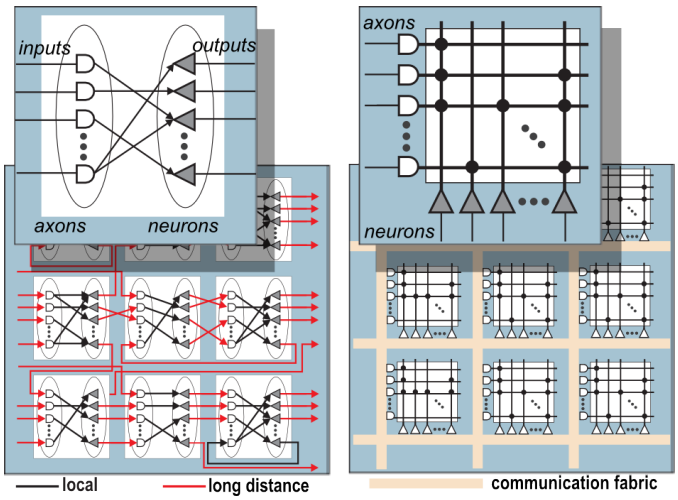


Fig. 2. The TrueNorth Architecture is based on a neurosynaptic core, tiled in a 2D array: logical representation (left) and physical implementation (right).

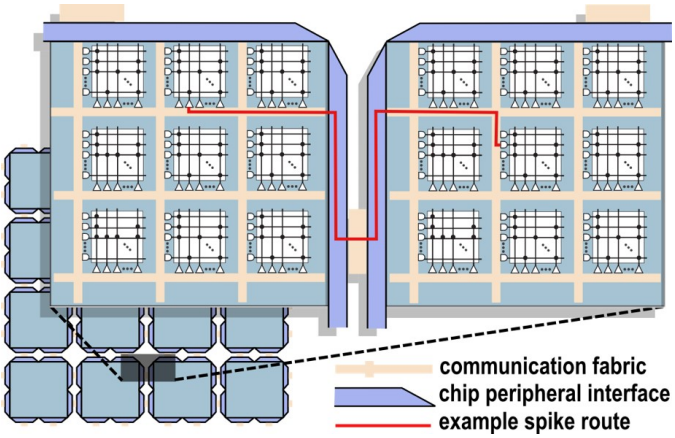


Fig. 3. An example spike route (red) between cores on two adjacent TrueNorth chips. Spikes traversing one row of the network-on-chip travel through the chip I/O peripheral circuitry and then emerge on the same row of the adjacent chip.

The scalability of the TrueNorth architecture derives from its modular tiled-core structure. Each TrueNorth *neurosynaptic core* represents a tiny fully-connected neural network with 256 output neurons and 256 input axons, connected by 256×256 synapses, one for every axon–neuron combination—a complete bipartite graph (Fig. 2 *upper left*). When cores are tiled on the chip and connected by a network-on-the-chip, these small bipartite graphs are combined to form a larger neural network. (Fig. 2 *lower left*).

The TrueNorth chip consists of $4,096$ cores, tiled as a 64×64 array, (Fig. 2 *lower right*). Each chip implements over 1 million neurons and over 256 million synapses, using 5.4 billion transistors fabricated in Samsung's 28nm LPP process technology. At 0.775V supply, each chip consumes approximately 70mW while running a typical vision application. Cores are implemented as a *fanout crossbar* structure (Fig. 2 *upper right*). An input spike event activates an axon, which drives all connected neurons. Neurons integrate incom-

ing spikes, weighted by synaptic strength. When a neuron membrane potential integrates beyond its threshold, it fires a spike, transmitting it to a preprogrammed target axon on any core in the network [9], [19], [20]. To support scaling beyond chip boundaries, chips can also be tiled in two dimensions via native event-driven SerDes links, enabling scaling to even larger networks, as shown in Fig. 3. This makes it possible and relatively simple to tile TrueNorth chips in a two-dimensional array, which is a critical property enabling the NS16e scale-up system presented here.

III. Hardware Systems

In this section, we present mobile, scale-out, and scale-up TrueNorth-based systems that form the building blocks to construct computing platforms and enable applications at multiple scales.

A. NS1e Platform

The **N**eurosynaptic **S**ystem, **1** million neuron **e**valuation platform (**NS1e**), shown in Fig. 4(a), supports embedded and mobile applications, and its compact form factor and modular nature make it a building block for scale-out systems of arbitrary size. In the embedded case, the NS1e can be deployed as a distributed supercomputer, pushing the computation close to the sensor. On the other hand, aggregating many NS1e boards together creates a centralized scale-out system, the same as the datacenter or cloud model.

The main processing element of the NS1e is a single TrueNorth chip (Fig. 4(d)). It is coupled on board with a Xilinx Zynq Z-7020 SoC (Fig. 4(e-f)) containing FPGA programmable logic and two ARM Cortex-A9 cores connected to 1GB DDR3 SDRAM. With the ARM cores running the Linux operating system and system software (Fig. 4(f)), as well as the FPGA performing data conversion and interface translation (Fig. 4(e)), this system can operate as a stand-alone system or a node in a scalable system. User interfaces include Gigabit Ethernet, micro USB, I2C, SPI, and UART, as well as three asynchronous spike-based interfaces directly to TrueNorth. Configurable GPIO provides an additional sensor/motor/actuator interface. The NS1e hosts the following on-board sensors: altimeter and thermometer (BMP-180), as well as: gyroscope, accelerometer, magnetometer and motion coprocessor (MPU-9150). The NS1e is 125mm$\times$69mm and weighs 98g. The average power consumption is between 2W and 3W (depending on the use case) with TrueNorth consuming only $\sim 3\%$ of the total power budget. The board can also operate in a full standalone mode while connected to a small Li-Po battery and transmit/receive data wirelessly, useful for autonomous applications.

B. NS1e-16 Platform

By connecting multiple NS1e boards via standard networking hardware, we created a scale-out system that runs many neural network instances in parallel. Fig. 4(b) shows the **NS1e-16** system constructed using sixteen NS1e boards, with aggregate capacity of 16 million neurons and 4 billion synapses, interconnected via a 1Gig-Ethernet packet switched network (Fig. 4(g)). The system optionally includes a traditional server (3.4 GHz quad-core Xeon, 32GB RAM, 1 TB hard drive) to act as a host gateway. The NS1e-16 system is integrated into a single 6U rack, and consumes a total of approximately 68W not including the server ($\sim$ 56W for the NS1e node cards and $\sim$ 12W for the network communication).

This system has the following properties, similar to other scale-out systems. First, computing capacity is increased by simply adding additional neurosynaptic compute nodes. Second, this approach has a reduced cost when compared to board-level integration. Third, board-level failures can be easily corrected by swapping out nodes. This platform is well suited for problems which can be decomposed and mapped onto a set of parallel and individual networks, each sized to the capacity of a single TrueNorth. Examples include an image recognition problem where a high resolution input image space is tiled across several boards, or combing through "big data" from a variety of sources or sensors.

C. NS16e Platform

Using TrueNorth's native chip-to-chip asynchronous communication interfaces, the **NS16e** (**N**eurosynaptic **S**ystem **16** million neuron **e**valuation) platform seamlessly integrates 16 TrueNorth chips into a *scale-up* solution (Fig. 4(c)). Present on all four sides of TrueNorth, these native inter-chip interfaces (Fig. 3) create a logically seamless grid of cores across different chips. On the NS16e board, a two-dimensional 4×4 array of TrueNorth chips are directly interconnected using these interfaces, creating a platform that is capable of executing neural networks 16 times larger than the NS1e.

The NS16e system consists of three boards assembled using vertical stacking/mating connectors: a custom 4×4 board (Fig. 4(h)), mated to a custom interposer board with an off-the-shelf Avnet 7Z045 Zynq SoC mini module (Fig. 4(i)). The interposer board provides high speed interfaces and power to all of the NS16e system components. It contains a PCIe x4 connector and an SFP+ Ethernet cage. The Zynq SoC mini module provides two ARM cores and an FPGA fabric for interface, control, and data manipulation. The ARM cores enable the NS16e to run as a standalone system. However, in the primary configuration, we disable the ARM cores and use the high-speed transceivers on the SoC as a PCIe communication bridge to a host computer (Dual Intel Xeon E5-2630 processors, 256GB RAM, and two Nvidia Titan-X GPU cards), shown in Fig. 4(j). This latter system configuration is a hybrid system combining neurosynaptic accelerators and traditional processor computation. For example, the CPU/GPU/FPGA may be used for pre-/post-processing or data transduction, such as applying image and audio filters or performing spike conversion, while the NS16e performs neural network execution. This system also supports a dual workflow in which neural network training is run on the host system GPU, and the main neurosynaptic run-time computation is carried out by the NS16e (see Section IV for more details).

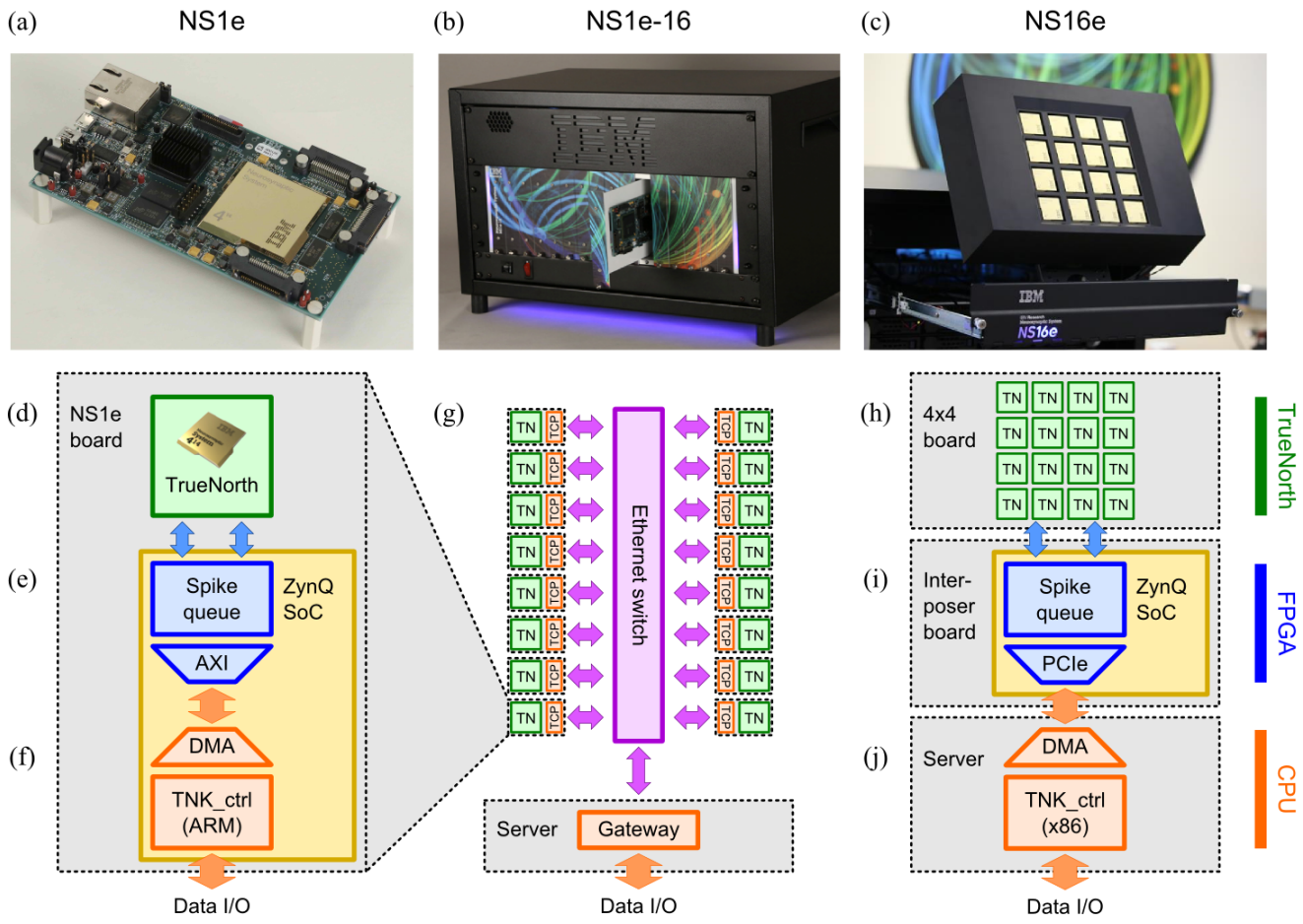


Fig. 4. Three neurosynaptic hardware systems: NS1e (a), NS1e-16 (b), NS16e (c), with respective system architecture below. See text for details.

D. Test and Verification

To improve the yield of TrueNorth based systems, defects are hidden at runtime so that the all systems behave, at the spike level, deterministically and in a one-to-one correspondence with the software simulator. The SRAMs in TrueNorth are designed with both row and column redundancy to repair cell defects. Defective SRAM cells are identified during testing and skipped at runtime, creating a transparent interface to the driving software and to the user. Using test reports that identify manufacturing defects in any of the 4,096 cores, a core map is used at runtime to physically place the neural model avoiding any imperfect cores.

TrueNorth also has a "debug spike" capability. Typically spikes exchanged between cores inside the chip are not visible from the outside, but duplicated copies of every normal spike can be sent out of the chip as debug spikes. By comparing them with the spikes computed by the simulator and locating the earliest discrepancy, we can identify which cores are malfunctioning. By disabling bad cores, the rest of the system can be restored to normal functionality.

E. Firmware

The firmware layer runs on each of the hardware platforms, consisting of FPGA programmable logic to efficiently bridge the *physical* interface to TrueNorth and a software driver to present a uniform *programming* interface across platforms.

When building neurosynaptic applications on our systems, the physical link between TrueNorth and a host system or external sensor often becomes the bottleneck. One goal of the firmware is to make this link efficient and high-throughput. The NS1e system interface is an AXI bus (Fig. 4(e)), while the NS16e uses PCIe 2.0 with a single lane (Fig. 4(i)). DMA engines with flow control speed up the transfer of spike, control, and configuration data over the bus interfaces. For the NS1e, this DMA engine is available as a part of the Zynq SoC (Fig. 4(f)), with generic Linux device driver code. For the NS16e, this logic was implemented using the PCI Express bus as the transport mechanism (Fig. 4(j)), with the latest PCI IOMMU technology and user space driver (Linux VFIO). Upgrading the firmware to use the DMA engines achieved a 5.25× and 3.90× throughput over memory mapped I/O

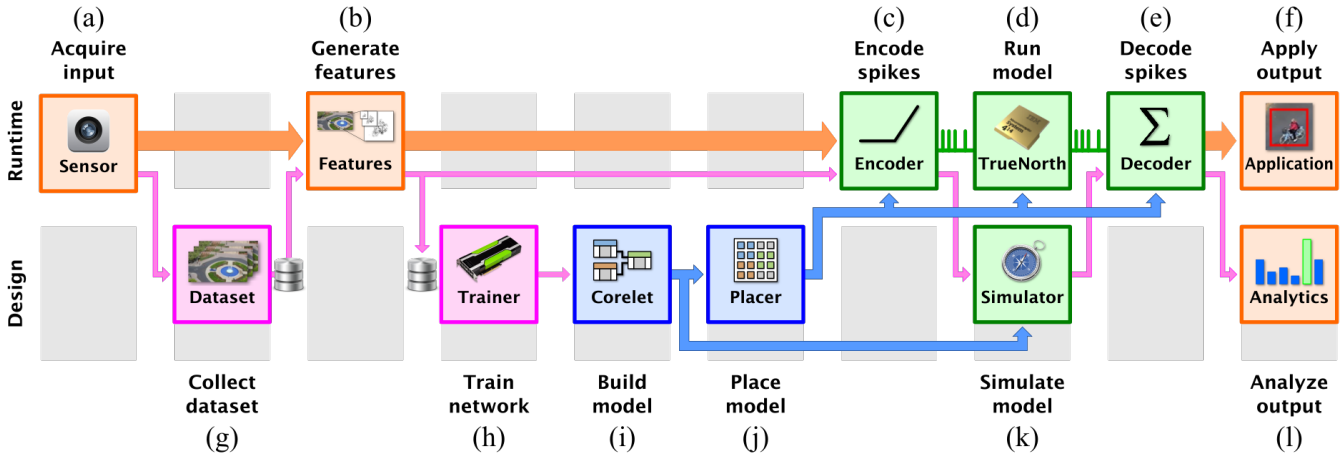


Fig. 5. Design workflow (bottom) and runtime workflow (top) for a TrueNorth application. Block colors correspond to: data pre-/post-processing (orange), training (magenta), network build (blue), and running the neural network (green). The design workflow is a superset of the runtime workflow, allowing training (magenta arrow) to coopt runtime components where feasible.

accesses on the NS1e and NS16e respectively.[1] The FPGA firmware implements spike queue logic (Fig. 4(e,i)) in order to 1) automatically send data to the port that minimizes the path length within the TrueNorth array, 2) dynamically load-balance between queues according to traffic, as well as to 3) convert spike data to/from the TrueNorth asynchronous chip interface protocol. The FPGA logic has also been used for data preprocessing tasks such as transduction from raw image/audio/video data to neural spike formats, or applying feature extraction and other front-end filtering transforms.

As an application interface, the TrueNorth Kernel Controller (TNK_ctrl) configures and streams data to/from all TrueNorth systems, abstracting away hardware details to hide the differences between platforms. TNK_ctrl reads a neural network model file and a physical core placement file from the software ecosystem tools (Section IV) and configures each TrueNorth chip. When loading the configuration data, TNK_ctrl accounts for any manufacturing defects such as bad cores or defective SRAM cells. After loading the neural model, TNK_ctrl controls the flow of data to/from the TrueNorth system, either through file I/O or streaming data via a networking protocol (TCP or UDP). TNK_ctrl runs natively on the NS1e's ARM cores (Fig. 4(f)), or on the NS16e's x86 host controller (Fig. 4(j)), providing the same interface in both cases so that applications run transparently on either system.

IV. Software Ecosystem

In this section, we present an end-to-end software ecosystem for specifying, training, building, and deploying neural network applications. Shown in Fig. 5, this set of TrueNorth development tools centers on two end-to-end workflows to *design* applications for TrueNorth, and to deploy those applications at *runtime*. These workflows operate identically on all TrueNorth hardware systems.

A. Runtime Workflow

At runtime (Fig. 5(a)–(f)), the basic structure of a generic TrueNorth application can be divided into six stages. First, a stream of input data is acquired from some source (Fig. 5(a)), which may be files read from disk, frames from a conventional sensor connected via USB or Ethernet, or even natively generated spikes from a neuromorphic sensor [21]. Next, the raw input data is often preprocessed (Fig. 5(b)) to reshape, crop, filter, or otherwise transform the signal; and to generate more specialized features like multi-scale edge maps. These generated features are then encoded as spike streams (Fig. 5(c)) for entry into TrueNorth, which processes input spikes according to the neural network model (Fig. 5(d)). TrueNorth produces a stream of output spikes that must be decoded (Fig. 5(e)) into semantically meaningful signals that are sent to the application layer (Fig. 5(f)) for visualization or other action.

To distribute the workflow seamlessly across the various hardware components of a TrueNorth system, we have developed a TrueNorth Runtime Workflow comprising a set of C++ APIs and applications that run on both embedded systems and Linux workstations. Depending on the task, the Runtime Workflow can perform feature generation and encoding in various locations, such as the host workstation, embedded ARM cores, FPGA, or intrinsically in a spiking sensor. Similarly, the spiking output can be decoded in multiple locations (embedded ARM, workstation CPU, or remote system).

B. Design Workflow: Overview

Designing a TrueNorth application means specifying the configuration of all six stages of the Runtime Workflow. The data pre-/post-processing stages (orange boxes in Fig. 5) are generic to any similarly structured non-TrueNorth application,

[1]Using two test programs, we measured this result by generating and transferring blocks of spike data between the host CPU and the TrueNorth chips via the FPGA using both memory mapped I/O and DMA methods. The performance reported uses DMA transfer sizes which reflect typical spike rates.

so we focus here on how to configure the spiking stages (green boxes): the encoder (Fig. 5(c)), decoder (Fig. 5(e)), simulator (Fig. 5(k)), and TrueNorth itself (Fig. 5(d)).

The Design Workflow (Fig. 5(g)–(l)) uses layered abstractions to insulate the developer from the complexity of the TrueNorth hardware. A core's configuration includes the axon type of each of its 256 axons, the output spike target and 23 neuron parameters for each of its 256 neurons, and the 256×256 binary synaptic crossbar connecting its axons to its neurons. The entire TrueNorth configuration is stored in a model file that is automatically generated by the Corelet Programming Environment (CPE), a MATLAB-based suite of tools that implement and support the Corelet Programming Language — a hierarchical, compositional language for programming core-based neuromorphic architectures [22] — that constitutes the first layer of abstraction.

Embedded in CPE, the Design Workflow has two phases: a *train* phase (Fig. 5(g)–(h)) that uses standard deep learning techniques to train a set of network parameters constrained to the TrueNorth architecture; and a *build* phase (Fig. 5(i)–(k)) that uses corelets to automatically compile a TrueNorth model from the network parameters, evaluates the logical model with the Compass simulator, and places each core in the logical model at a physical location in hardware.

C. Design Workflow: Train Step

1) Dataset: The starting point for any supervised learning task is a set of labeled training data that is representative of the signal stream to be processed, preferably collected (Fig. 5(g)) using the intended input device (Fig. 5(a)) and preprocessed into the actual features (Fig. 5(b)) that will be presented to the classifier at runtime (Fig. 5(c)–(d)). We import this data into an efficient, multi-dimensional, multi-channel format that spans a wide range of sensor modalities and can be read by all our downstream tools. Data is stored in a Lightning Memory-Mapped Database (LMDB), a high-performance, embedded, transactional database that is a preferred input format for many deep learning frameworks due to its fast read access.

2) Trainer: The Design Workflow packages two different algorithms for training a TrueNorth classifier on a labeled dataset (Fig. 5(h)). Each algorithm abstracts low-level TrueNorth design constraints — such as neurons that use spikes instead of continuous values, synapses with low precision instead of high precision, and connectivity that is core-to-core instead of all-to-all — into more familiar deep learning concepts — such as convolution kernel size, neurons per layer, and learning rate — that we implement in standard GPU-accelerated training frameworks such as Caffe or MatConvNet.[2] Recent algorithmic innovations have shown the power of deep learning for low-precision computing [11], [23].

The first algorithm employs a version of backpropagation that trains in a probabilistic domain with a direct correspondence to the TrueNorth hardware. This algorithm is implemented in the Caffe framework, an open-source C++ library for backpropagation-based training of deep neural networks [2]. This approach has demonstrated near state-of-the-art performance on the MNIST dataset, while performing 1000 classifications per second at 108μJ per image [12].

The second algorithm is an adaptation of deep Convolutional Neural Networks (CNNs), that constrains the network to spiking neurons, trinary weights $\{-1, 0, 1\}$, and the core-to-core connectivity of the TrueNorth architecture. The trinary weights are used during the forward and backpropagation passes of offline training. However, weight updates are each applied to a high precision shadow weight, from which a trinary weight is derived by rounding with hysteresis. That is, the trinary weight is updated only when the shadow weight accumulates more than sufficient weight changes. Neuron firing is determined by a step function, which does not have a finite derivative required to support backpropagation. To avoid this problem, our algorithm approximates the derivative with a triangle function. Additionally, the TrueNorth connectivity constraints are applied when constructing the network to train, requiring that all convolutional filters are able to fit onto a single TrueNorth core. As a result, the trained network can be directly mapped to TrueNorth with no loss in task accuracy. The above algorithm is implemented using custom MATLAB code, with acceleration for convolutional operations provided functions from the MatConvNet library [3]. This method was used to approach state-of-the-art accuracy across 8 image and audio datasets, with measured maximum throughput of the chip of $1,200$ to $2,600$ frames per second while using between 25 and 275 mW [13].

These high-level frameworks free a designer to focus on abstract network properties instead of configuring TrueNorth neurons and their connectivity directly. The output of the trainer is a set of learned weights and parameters that are directly ingested by the corelet stage of the build phase (Fig. 5(i)) to compile a hardware-ready TrueNorth model.

D. Design Workflow: Build Step

1) Corelet and Simulator: The model building process, shown in Fig. 5(i), is an automatic compilation from a high-level description of the network to a data format readily consumable by the TrueNorth hardware. This process uses a library of standard corelets (self-contained modules of TrueNorth cores and connectivity required to implement a particular function [22]). Trained classifiers may also be combined with front-end and back-end functional blocks like pre-processing filters and feature extractors or post-processing smoothing and aggregation functions. These user-specified corelets are combined with the classifier corelet and compiled into the set of TrueNorth model files in the build step.

The result of the model building process is used to configure the spike encoder (Fig. 5(c)), decoder (Fig. 5(e)), the Compass simulator (Fig. 5(k)), and after an additional placement step (Fig. 5(j)), physical TrueNorth hardware (Fig. 5(d)). The Compass simulator [17], written in C++ with OpenMPI, is a highly parallel implementation of the TrueNorth specification

[2]The choice of training framework is incidental to the algorithms, which could be ported to other frameworks like Theano, Torch, or TensorFlow.

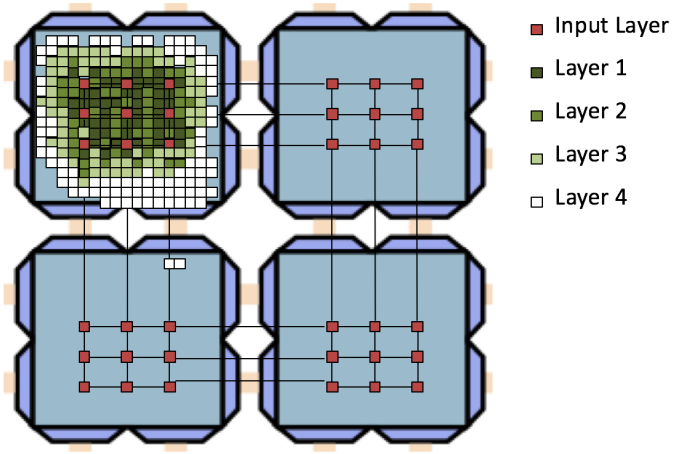


Fig. 6. Neurosynaptic core placement algorithm example illustration. Following input space partitioning and placement, cores are placed layer-by-layer minimizing the total path cost over core inputs. (Layer placement only shown for one chip.)

and spike-for-spike equivalent to actual TrueNorth hardware. A separate branch from the main workflow, it is used for early application development before hardware availability, as well as system and application debugging.

2) Placer: The final stage of the Design Workflow (Fig. 5(j)) maps logical cores in the TrueNorth model to physical cores in TrueNorth hardware, much like a VHDL program is compiled into an FPGA bitfile. We developed a heuristic placement optimization tool that efficiently addresses this NP-hard placement mapping challenge. Given a network model and the target hardware chip array, the placer attempts to find a core mapping that minimizes inter-chip communication, minimizes total spike travel distance, and maximizes external input and output throughput to support the very high frame rates achieved in Sec. IV-C.

The active energy required to route a spike between two cores is a function of the grid locations of the two cores. Specifically, the total path cost is the sum of the internal (intra-chip) communication energy cost and the inter-chip communication energy cost (Figure 3). Spike travel across chip boundaries is significantly more expensive in terms of bandwidth and energy than travel within the on-chip routing network, so while a good placement algorithm has limited impact on single-chip systems, it is essential for multi-chip systems. Total path cost minimization is achieved by maximizing the number of paths that are completely contained within one chip, and minimizing the number of paths that require inter-chip hops. This problem of mapping neurosynaptic cores to minimize spike communication power is similar to the wire-length minimization problem in VLSI placement [24]. However, simpler algorithms work well when the network graph is topographically well-structured.

We developed a heuristic placement algorithm that leverages the topographic connectivity structure in TrueNorth networks developed in both the Caffe and MatConvNet frameworks, as

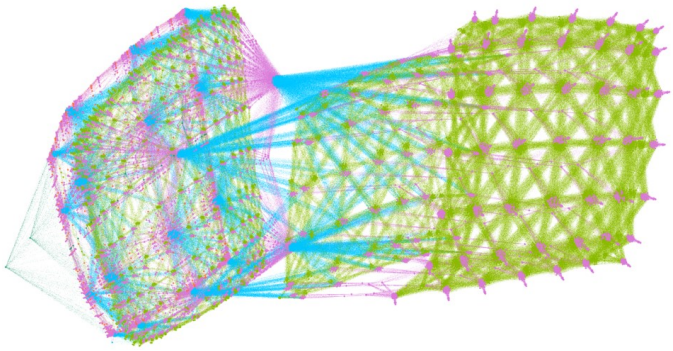
Algorithm 1 Algorithm for Core Placement

Require: Graph $G = (V, E)$ where
$v \in V$ is a core and
$e \in E$ is a connectivity between cores
V_I: set of input cores
K: number of chips to be mapped

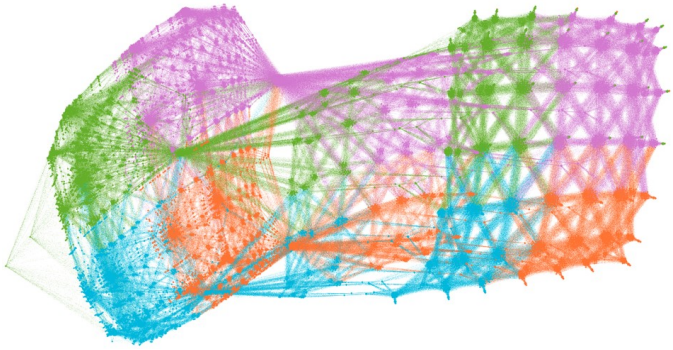
Procedure: NeuroSynapticCorePlacer
Mark location (chip, x, y) of each defective core unavailable.
Step A: Preprocessing Step
for $v \in$ V **do**
$V^{in}(v) \leftarrow$ list of inward adjacent vertices of v
end for
Step B: Input Partitioning
Create K clusters C_k of vertices $v \in V_I$ so that each cluster C_k has approximately $\frac{|V_I|}{K}$ cores, such that each cluster C_k receives input from a topographically contiguous input region.
Step C: Input Placement
for $k \leftarrow 1$ to K **do**
for core $v \in C_k$ **do**
Algorithmically place v inside chip_k.
end for
end for
Step D: Place Cores By Layer
Let D be the maximum depth of graph G
Let ${v_i}^d$ denote the i^{th} core at depth d
for $d \leftarrow 1$ to $(D-1)$ **do**
for each core ${v_i}^d$ **do**
Select k to maximize $|V^{in}(v_i^d) \cap C_k|$.
(If chip_k has no available core, choose another k.)
$C_k \leftarrow C_k \cup \{v_i^d\}$.
Compute available location (chip_k, x, y) to minimize the sum of Manhattan distances between v_i^d and $v \in V^{in}(v_i^d)$ with reference to the current placement.
Place ${v_i}^d$ to (chip_k, x, y) and mark it unavailable.
end for
end for

described in Section IV-C. The pseudocode of the algorithm is included in Algorithm 1. In Step B to Step C, the algorithm clusters and co-locates cores that process topographically neighboring regions of the input signal first. These are the *Input layer* cores shown in Fig. 6. Then cores from each consecutive network layer are placed iteratively in the same chip as their source cores while minimizing the cost function (Step D). An example of this process is illustrated in Fig. 6. This technique generates placement solutions much faster than an earlier algorithm [20], while sufficiently reducing inter-chip communication, as shown in Figure 7(b).

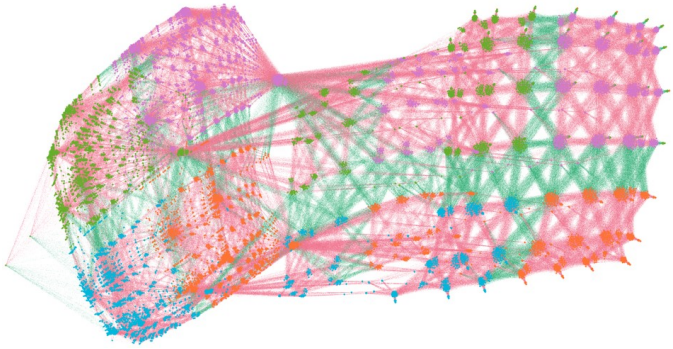
To illustrate the result, we trained a CNN on the CIFAR-10 visual object recognition dataset, resulting in a 4-chip network with 15,138 neurosynaptic cores. Figure 7(a) uses Gephi [25], an open-source graph analysis tool, to plot the network's core-to-core connectivity graph before placement. The network



(a)



(b)



(c)

Fig. 7. Core-to-core connectivity graph for a 4-chip, 21-layer CIFAR-10 network. Input layers are to the left; output layers are to the right. (a) Cores belonging to convolution, pooling, and splitter layers are colored with magenta, blue, and green, respectively. (b) Cores and connections are colored (blue, green, purple, red) according to the chip on which they are placed. (c) Intra-chip connections are colored red, inter-chip connections are colored green, and cores (graph nodes) are colored according to chip.

contains 13 convolution layers and 2 mean-pooling layers, consuming approximately 69% and 6% of the cores, respectively. In addition, 6 splitter layers generate the duplicate inputs required for the overlapping kernels of the convolution and pooling layers, consuming another 25% of the cores. Observe that the topographic connections between layers allows for efficient local grouping of neurosynaptic cores, which in turn enables the network to be efficiently mapped onto the chips with minimal inter-chip communication, as shown in Figures 7(b) and (c). With this optimized placement, 84.66% of all

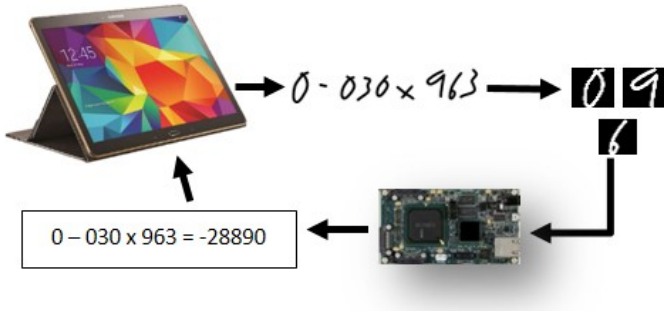


Fig. 8. Dataflow of the interactive handwritten character recognition application. A user hand-writes a mathematical formula on a tablet; the characters are segmented and sent to the NS1e for recognition. Classification results are sent back to the tablet which completes the symbolic processing, returning the answer to the equation.

connections are completely contained on the same chip and the remaining 15.34% are inter-chip connections.

V. Applications

We have a diverse group of motivated early adopters, including 8 government agencies, 3 national labs, 7 corporate research centers, and 25 universities. These initial ecosystem partners underwent 3 weeks of in-depth training at a "Boot Camp" in the summer of 2015 and a "Boot Camp Reunion" workshop in the spring of 2016. Additional early adopters came from 2015 Telluride Neuromorphic Engineering Workshop. Participant selection was based on expertise in neuromorphic systems and application of neural networks, in areas with potential to leverage TrueNorth's low power, speed, flexibility and scalability. Training encompassed all facets of the end-to-end ecosystem, beginning with basic architecture and building blocks such as the neuron model, synaptic crossbar, and fundamentals of efficient neurosynaptic computation. In the 2016 workshop, the participants went through training on the Software Ecosystem Design Workflow (Section IV-B), and the majority of them could build, train, and run deep convolutional networks on new datasets on the NS1e over the course of 1.5 days. Under a hardware loan and software evaluation agreement, ecosystem partners have continued to pursue work with TrueNorth at their home labs in the application areas described in Table I. This comprehensive software ecosystem has also been used to achieve near state-of-the-art results on 8 image and audio datasets [13].

Following are three specific example applications currently under investigation, using the end-to-end ecosystem, that are relevant to the scientific community.

A. Example NS1e Application: Interactive Handwritten Character Recognition (Army Research Laboratory)

Real time computing at low power makes it feasible to develop a suite of sensors with tightly integrated cognitive computing capabilities at the data collection site. Toward this end, we have developed a computational offloading scheme that collects data from a camera or a tablet, processes it using

TABLE I
ECOSYSTEM PARTNER PROJECTS.

Multimodal Sensory & Signal Processing	
Hyperspectral, Radar, & Image Classification	Air Force Research Lab Lawrence Livermore National Lab Riverside Research SRC Inc. TSC Inc.
Supernova Detection	Lawrence Berkeley National Lab
Generative Inference & Generative Models	UC San Diego
Sparse Approximation	Georgia Tech
Motion Energy	UC Irvine
Video Tracking	University of Wisconsin
Perceptive Ultrasound	University of Western Ontario
Embedded & Autonomous Systems	
Robotics	Army Research Lab Imperial College Johns Hopkins - Applied Physics Lab Navy Research Lab Technical University of Munich
Asynchronous Circuits	Cornell University
Native Spiking Sensor Integration	
Event-based Optical Flow	Ulm University
Direct Spiking Vision Sensor Interface	Nanyang Technical University National University of Singapore
Real-time Sensory Information Processing	Johns Hopkins University
Neural Circuit Modeling	
Joint Feedforward Excitation & Inhibition	Arizona State University
Gated Information Routing	UC Davis
Canonical Cortical Circuits	INI/ETH Zurich
Spike-based Recurrent Network	INI/ETH Zurich
Learning & Optimization	
Evolutionary Optimization	Argonne National Lab University of Tennessee
Inference Accuracy	University of Pittsburgh
High Performance Computing	
HPC Node Failure Prediction	RPI
Autonomous Agents	Dayton
Adaptive Simulation	Lawrence Livermore National Lab
High Energy Physics & Particle/Event Detection	
Event Detection	Argonne National Lab Lawrence Berkeley National Lab
Skyrmion Track Detection	UCLA
Text, Audio & NLP	
Probabilistic Inference	Air Force Research Lab Syracuse University
Low-Power Audio Transform	Penn State University
Sentiment Analysis	INI/ETH Zurich
Prosody	UC Santa Cruz

TrueNorth, and sends the results to a remote location for situational awareness or further conventional processing. Using the software ecosystem and specifying the neural networks in the Corelet Programming Language, Fig. 5(i), we produced

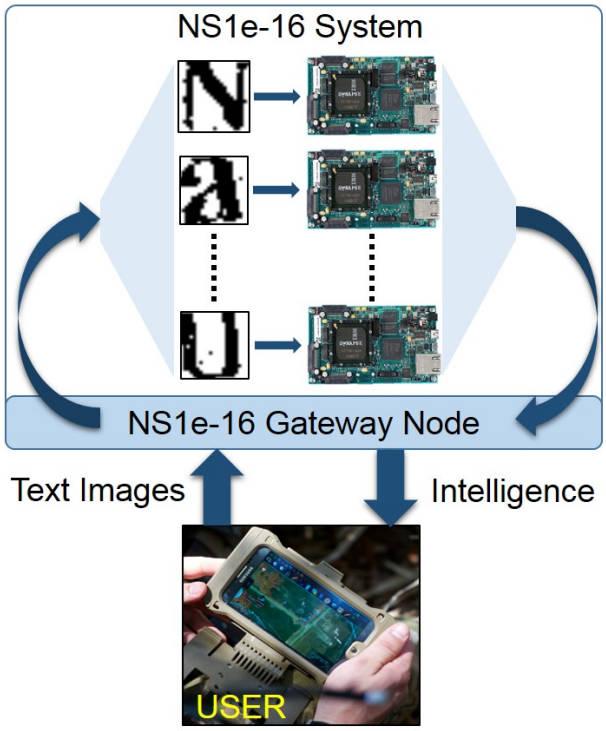


Fig. 9. Data-Parallel Text Extraction and Recognition System: character classification workload is distributed over 16 nodes in the NS1e-16 system.

an interactive handwritten character recognition demo for visitors to the laboratory. They can hand write an arithmetic expression, such as: "$0 - 030 \times 963$," on an Android tablet (Fig. 8), and upon completion of each handwritten character, the canvas image is wirelessly transmitted to TrueNorth where each segment is classified into the appropriate digit or operator. The results are sent back to the tablet where the result is calculated and displayed in real time.

Looking forward, we will be exploring a tight coupling between three separate components: a) a mobile agent capable of sensing, navigating, manipulating objects, and interacting with people in the environment; b) the scale-out NS1e-16 system for fast, low-power processing of data collected by the mobile agent; and c) large supercomputer systems for time-critical retraining of new information and abilities captured by the mobile agents. The training systems use swarm-based distributed training approaches for fast multi-modal deep learning of vision, audio, and speech data. Combining these three system components produces a multi-faceted and multimodal cybernetic system that can act on data coming from the environment, capture novel encounters, and adapt to them for future encounters.

B. Example NS1e-16 Application: Data-Parallel Text Extraction and Recognition (Air Force Research Laboratory)

Processing of huge amount of raw data, especially noisy unclear images, printed documents, signs and other textual media, poses a serious challenge if the power budget is limited.

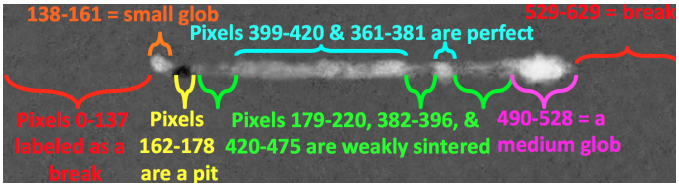


Fig. 10. Seven classes of defects shown in a sample image of a stainless steel track weld created by a Selective Laser Melting Additive Manufacturing process.

In order to demonstrate NS1e-16's scale-out capability of handling large amount of ambiguous data, we developed a text extraction application that distributes raw textual images across a NS1e-16 system and processes them in parallel (Fig. 9). First, a user submits a photo of the textual media to the NS1e-16 system's gateway node. The page image is then enhanced and segmented into individual characters which are streamed in parallel to the 16 TrueNorth processors for character recognition. The classification outputs are then sent to an inference-based natural language model that reconstructs the words and sentences in the page image. This data can then be queried by the user, unlocking all of the information contained in the documents.

In order to build this system, we trained CNN models for character recognition, using the TrueNorth ConvNet trainer (Fig. 5(h)). The input characters to the CNNs are 22×22 pixel black-and-white images, distributed by the preprocessing procedure running on the NS1e-16 gateway node. The CNN model consists of four convolutional layers, four max pooling layers (one following each convolutional layer), and three network-in-network layers. The entire model uses $2,370$ TrueNorth cores, and the CNN outputs classification scores for each of the 93 possible character classes. As to the recognition performance, we can classify up to $1,000$ characters per second per board, using a one-tick encoding scheme for the character images. During run-time, an identical network model is independently deployed to each node of the NS1e-16 system that runs the CNN classifier, achieving an aggregate throughput of $16,000$ characters per second. When a user submits a document image, the character images are extracted and distributed to each node in parallel using a TCP streaming protocol, interfacing directly with the TNK_ctrl firmware (Section III-E). While the current development performs only character recognition on TrueNorth, we are working toward implementing the word and sentence inference algorithms on TrueNorth using stochastic spiking neural network models.

C. Example NS1e Application: Defect Detection in Additive Manufacturing (Lawrence Livermore National Laboratory)

Additive manufacturing (AM) describes a host of fabrication technologies wherein a set of design and processing instructions are fed to a system that builds objects in a layer-by-layer fashion[26]. Existing state-of-the-art AM control systems do not allow for the build instructions to be adjusted according to *in situ* process monitoring data collected by sensors, e.g. high speed cameras, pyrometers, etc. The motivation for this work is to construct low-power systems for the automated detection and rectification of manufacturing defects, necessary for rapid build qualification, tighter tolerances, and higher yields for AM technologies.

Here we show that TrueNorth can be used to detect defects using imagery of stainless steel track welds, from a Selective Laser Melting (SLM) AM process. At 75 sites, a confocal image of size 170×629 is taken that details surface roughness along the length of the track. As shown in Figure 10, surface roughness is used to label each column of an image according to one of 7 possible classes: {break, big glob, medium glob, small glob, pit, weak sinter, and perfect}. The images are preprocessed by mapping from *RGB* to grayscale and transforming the pixels into the range [-1,1]. Each image is transformed into 629 samples by taking sub-images of size 50×5. The label for the sample comes from the center column of the sub-image.

We constructed a baseline five layer CNN (2 conv + ReLU, 2 fully connected, 1 softmax output) using Theano[27]. Using full floating-point precision, this model has an accuracy of 81% on our SLM-AM dataset. The TrueNorth model has a 10 convolutional layers using $3,348$ neurosynaptic cores and maps directly onto TrueNorth. This discrete, binary TrueNorth network has an accuracy of 80%.

VI. Power Measurements of NS1e and NS16e

In order to compare the power performance of our systems, we measured the power consumption of the NS1e and NS16e while running the CIFAR10 and CIFAR100 image recognition tasks using three sizes of deep convolutional networks (Table. II). The NS1e-16 system is built from 16 NS1e cards, and so the power can be inferred from the NS1e power. The NS1e system power consumes 3.17W when idle and 3.52W running these tasks. The NS16e system consumes 7.68W when idle and up to 8.88W for an 8 chip application. With a 1.0V power supply, TrueNorth chips consume 0.114W per chip when idle or $\sim$ 0.228W when active, depending on the network activity.

These results show the accuracy versus energy tradeoff inherent in our systems. Accuracy and energy efficiency, both functions of area (network size), are inversely correlated. Thus, finding the optimal network size for a particular task depends on the application requirements. If the energy efficiency is important, a smaller network model may be run on a single chip on the NS1e. For higher recognition accuracy, larger networks can be run on the NS16e. For recognition of larger images, we used the entire 16-chip array to do the task[10].

VII. Related Work

In recent years, a few groups have proposed power-efficient hardware for the emulation of biologically-inspired neural networks. Neurogrid modeled 1 million neurons across 16 chips in real time using mixed analog/digital signals and off-chip memory to store connectivity parameters [28]. The 48-node SpiNNaker system is based on an SoC containing 18

TABLE II
POWER, PROCESSING SPEED AND ACCURACY OF NS1E AND NS16E

Task	System	# Active Chip	Throughput (FPS)	System Power (W)	TrueNorth Power (W)	TrueNorth Energy Efficiency (FPS/W)	Accuracy
Idle Power	NS1e	0		3.17	0.114 (1 chip)		
	NS16e	0		7.68	1.824 (16 chips)		
CIFAR10	NS1e	1	1249	3.52	0.204 (1 chip)	6109	83.41%
	NS16e	4	1324	8.64	0.936 (4 chips)	1414	87.97%
	NS16e	8	1040	8.88	1.497 (8 chips)	695	89.00%
CIFAR100	NS1e	1	1526	3.52	0.208 (1 chip)	7344	55.64%
	NS16e	4	1257	8.64	0.891 (4 chips)	1410	63.86%
	NS16e	8	432	8.28	1.192 (8 chips)	362	66.32%

ARM cores, 128 MB of off-die memory, and a packet router with 6 bi-directional links [29]. Server racks contain 120 cards (48 SpiNNaker chips per card), and a cabinet of processors can model up to 100 million simplified point-neuron models. Neural network descriptions can be specified using the Python library PyNN, and loaded onto the hardware using lower-level driver software. The BrainScalesS Project is a mixed-signal architecture that combines analog neurosynaptic computation (512 neurons per chip), 128k SRAM based synapses, and digital asynchronous communication[30], [31]. The project combines 450 chips on a 20cm wafer into a single integrated system, and is programmed using PyNN.

For a rough comparison, we report the performance of five digital computational architectures running deep neural networks. The analog approaches above do not report performance results on state-of-the-art benchmarks. The manufacturing variability inherent in these approaches hinders the programmability and capability to have the viable software ecosystem required for large-scale neural network applications. A single TrueNorth chip processes $1200-2600$ 32×32 color images per second, consuming $170-275$mW, yielding an energy efficiency of $6100-7350$ FPS/W [13]. Current multi-chip TrueNorth networks process 32×32 color images at $430-1330$ per second, and consume $0.89-1.5$W, an energy efficiency of $360-1420$ FPS/W. SpiNNaker yields 167 FPS/W, processing 28×28 grayscale images [32]. Tegra K1 GPU, Titan X GPU, and Core i7 CPU yield 45, 14.2, and 3.9 FPS/W, respectively, processing 224×224 color images [33].

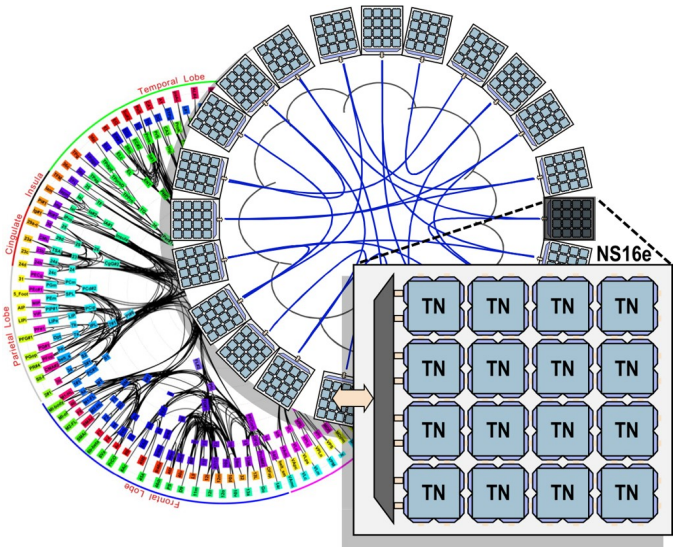


Fig. 11. Long-distance communication defined by the inter-regional connectivity in the macaque cortex [14] (left) and emulated by neurosynaptic systems (right). The neurosynaptic system is composed of densely connected neural regions instantiated on NS16e boards, interconnected via standard networking infrastructure (represented by the cloud).

VIII. VISION FOR FUTURE SCALED SYSTEMS

Looking to the future, the key to a scalable system architecture is hierarchical communication using native spike events at all levels, from the core to the chip to the board to the rack. Within a core, similar to communication within a cortical column, the synaptic crossbar (Fig. 2) delivers all-to-all connectivity between 256 input axons and 256 output neurons, achieving high bandwidth with minimal energy consumption. Between cores, similar to regional cortical connectivity, the asynchronous routing network enables very efficient intra-chip (Fig. 2) and inter-chip (Fig. 3) mid-range communication whose energy consumption scales with the immediate bandwidth requirement. Between boards, similar to long-range cortical connectivity, TrueNorth systems will leverage available networking infrastructure, such as Ethernet, Infiniband, or PCIe, as depicted in Fig. 11. This scalable and hierarchical routing architecture is possible due to the exponential decay in hop distance and bandwidth observed in biological networks [14]. More importantly, this distribution is similar to that of the hop-length histograms of placed deep-networks that are run on TrueNorth.

Based on the energy-efficiency of TrueNorth and the principles of hierarchical communication, we are developing large-scale neurosynaptic systems. These systems are constructed by integrating NS16e boards with commercially available networking solutions. We envision these systems scaling up to hundreds of racks, with billions of neurons and trillions of synapses, that run complex, integrated, large-scale neural applications with unprecedented energy-efficiency and throughput. We envision that future brain-inspired processors, systems, and software – powered by innovations from the scientific community – will become a key component in exascale systems.

ACKNOWLEDGMENT

We thank Charles Cox, Mike Criscolo, and Ken Inoue for their contribution in building the NS16e scale-up system, Peter Carlson, Guillaume Garreau, Jen Klamo, Jeffrey Mckinstry, and Burak Yildiz for their contribution in building the TrueNorth Ecosystem, and Gi-Joon Nam for providing insights into the core placement problem. We also thank Richard Vuduc for carefully shepherding the paper and suggesting many improvements. Heinz Baier, Tom Hovarth, and James Infantolino are employees of Achieve-it, LLC, L.J. Gonzer Associates, and Engility Corporation, respectively. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the U.S. Government or any agencies thereof.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[3] A. Vedaldi and K. Lenc, "MatConvNet – Convolutional Neural Networks for MATLAB," in *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, pp. 1–42, 2014.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[7] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "Strategies for training large scale neural network language models," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2011, pp. 196–201.

[8] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*. ACM, 2008, pp. 160–167.

[9] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[10] A. Cassidy, R. Alvarez-Icaza, F. Akopyan, J. Sawada, J. Arthur, P. Merolla, P. Datta, M. Gonzalez-Tallada, B. Taba, A. Andreopoulos *et al.*, "Real-Time Scalable Cortical Computing at 46 Giga-Synaptic OPS/Watt with $\sim 100\times$ Speedup in Time-to-Solution and $\sim 100{,}000\times$ Reduction in Energy-to-Solution," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.

[11] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3105–3113.

[12] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in Neural Information Processing Systems*, 2015, pp. 1117–1125.

[13] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *arXiv preprint arXiv:1603.08270*, 2016.

[14] D. S. Modha and R. Singh, "Network architecture of the long-distance pathways in the macaque brain," *Proceedings of the National Academy of Sciences*, vol. 107, no. 30, pp. 13 485–13 490, 2010.

[15] R. Ananthanarayanan and D. S. Modha, "Anatomy of a cortical simulator," in *Supercomputing 07*, 2007.

[16] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. IEEE, 2009, pp. 1–12.

[17] R. Preissl, T. M. Wong, P. Datta, M. Flickner, R. Singh, S. K. Esser, W. P. Risk, H. D. Simon, and D. S. Modha, "Compass: A scalable simulator for an architecture for cognitive computing," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 54.

[18] T. M. Wong, R. Preissl, P. Datta, M. Flickner, R. Singh, S. K. Esser, E. McQuinn, R. Appuswamy, W. P. Risk, H. D. Simon *et al.*, "10^{14}," *IBM Research Divsion, Research Report RJ10502*, 2012.

[19] A. S. Cassidy, P. Merolla, J. V. Arthur, S. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman *et al.*, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *International Joint Conference on Neural Networks (IJCNN). IEEE*, 2013.

[20] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, Oct 2015.

[21] T. Delbrück, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2010, pp. 2426–2429.

[22] A. Amir, P. Datta, A. Cassidy, J. Kusnitz, S. Esser, A. Andreopoulos, T. Wong, W. Risk, M. Flickner, R. Alvarez-Icaza *et al.*, "Cognitve computing programming paradigm: A corelet language for composing networks of neuro-synaptic cores," in *International Joint Conference on Neural Networks (IJCNN). IEEE*, 2013.

[23] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *arXiv preprint arXiv:1603.05279*, 2016.

[24] J. Cong and G.-J. Nam, "Modern circuit placement: best practices and results," *Springer Verlag*, 2007.

[25] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *International AAAI Conference on Weblogs and Social Media*, 2009.

[26] C. Kamath, "Data mining and statistical inference in selective laser melting," *The International Journal of Advanced Manufacturing Technology*, pp. 1–19, 2016.

[27] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010, oral Presentation.

[28] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," in *Proceedings of the IEEE*, 2014, pp. 699–716.

[29] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker Project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[30] K. Meier, "A mixed-signal universal neuromorphic computing system," in *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2015, pp. 4.6.1–4.6.4.

[31] J. Schemmel, D. Bruderle, A. Grubl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *International Symposium on Circuits and systems (ISCAS)*. IEEE, 2010, pp. 1947–1950.

[32] E. Stromatias, D. Neil, F. Galluppi, M. Pfeiffer, S.-C. Liu, and S. Furber, "Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on SpiNNaker," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8.

[33] "GPU-based deep learning inference: A performance and power analysis," Whitepaper, nVidia, Nov. 2015.