Cognitive Computing Systems: Algorithms and Applications for Networks of Neurosynaptic Cores

Steve K. Esser, Alexander Andreopoulos, Rathinakumar Appuswamy, Pallab Datta, Davis Barch, Arnon Amir, John Arthur, Andrew Cassidy, Myron Flickner, Paul Merolla, Shyamal Chandra[§], Nicola Basilico[†], Stefano Carpin,[†], Tom Zimmerman, Frank Zee[§], Rodrigo Alvarez-Icaza, Jeffrey A. Kusnitz, Theodore M. Wong, William P. Risk, Emmett McQuinn, Tapan K. Nayak[‡], Raghavendra Singh[‡], and Dharmendra S. Modha
IBM Research - Almaden, San Jose, CA 95120 [‡]IBM Research - India [†]UC Merced, Merced, CA 95343

***Abstract*—Marching along the DARPA SyNAPSE roadmap, IBM unveils a trilogy of innovations towards the TrueNorth cognitive computing system inspired by the brain's function and efficiency. The non-von Neumann nature of the TrueNorth architecture necessitates a novel approach to efficient system design. To this end, we have developed a set of abstractions, algorithms, and applications that are natively efficient for TrueNorth. First, we developed repeatedly-used abstractions that span neural codes (such as binary, rate, population, and time-to-spike), long-range connectivity, and short-range connectivity. Second, we implemented ten algorithms that include convolution networks, spectral content estimators, liquid state machines, restricted Boltzmann machines, hidden Markov models, looming detection, temporal pattern matching, and various classifiers. Third, we demonstrate seven applications that include speaker recognition, music composer recognition, digit recognition, sequence prediction, collision avoidance, optical flow, and eye detection. Our results showcase the parallelism, versatility, rich connectivity, spatio-temporality, and multi-modality of the TrueNorth architecture as well as compositionality of the corelet programming paradigm and the flexibility of the underlying neuron model.**

I. Introduction

A. Context

To usher in a new era of cognitive computing [1], we are developing TrueNorth (Fig. 1), a non-von Neumann, modular, parallel, distributed, event-driven, scalable architecture—inspired by the function, low power, and compact volume of the organic brain. TrueNorth is a versatile substrate for integrating spatio-temporal, real-time cognitive algorithms for multi-modal, sub-symbolic, sensor-actuator systems. TrueNorth comprises of a scalable network of configurable neurosynaptic cores. Each core brings memory ("synapses"), processors ("neurons"), and communication ("axons") in close proximity, wherein inter-core communication is carried by all-or-none spike events, sent over a message-passing network.

Recently, we have achieved a number of milestones: first, a demonstration of 256-neuron, 64k/256k-synapse neurosynaptic cores in 45nm silicon [2], [4] that were featured on the cover of *Scientific American* in December 2011; second, a demonstration of multiple real-time applications [5]; third, Compass, a simulator of the TrueNorth architecture, which simulated over 2 billion neurosynaptic cores exceeding 10^{14} synapses [3], [6]; and, fourth, a visualization of the long-distance connectivity of the Macaque brain [7]—mapped to TrueNorth architecture—that was featured on the covers of *Science* [8] and *Communications of the ACM* [1].

We now unveil a series of interlocking innovations in a set of three papers. In this paper, we present a set of algorithms and applications that demonstrate the potential of the TrueNorth architecture and value of the programming paradigm. In two companion papers, we present a versatile and efficient digital spiking neuron model that is a building block of the TrueNorth architecture [9] as well as a programming paradigm for hierarchically composing and configuring cognitive systems that is effective for the programmer and efficient for the TrueNorth architecture [10].

B. Motivation

We live in a world where a rapid proliferation of sensors, embedded in a range of consumer devices like cars, cameras, and cell phones as well as complex systems like weather monitoring networks, provide a massive flow of real-time, spatio-temporal, multi-modal data. The volume, the velocity, the variety, and the veracity of the data all pose enormous computational, algorithmic, power, and system packaging challenges. To complement today's computational paradigm that brings *data to computation* and excels at *symbolic* processing, we envision a TrueNorth-based computational paradigm that brings *computation to data* and excels at *sub-symbolic* processing. Specifically, the low-precision, synthetic, simultaneous, pattern-based metaphor of TrueNorth is a fitting complement to the high-precision, analytical, sequential, logic-based metaphor of today's von Neumann computers.

A TrueNorth *program* is a complete specification of the anatomy and physiology of a network of neurosynaptic cores that identifies all external inputs and outputs of the network. Abstractions, algorithms, and applications that are efficient for the von Neumann architecture are not necessarily efficient for the non-von Neumann TrueNorth architecture. Therefore, we are unable to directly leverage the rich heritage of design, development, and deployment of the former and instead are compelled to seek a new way of thinking that is custom tailored to the latter. Specifically, to write good TrueNorth programs, three concrete challenges must be overcome. First, one must learn to program a parallel, distributed, event-driven architecture that uses spikes, has discrete-valued synapses, allows each neuron to receive inputs from at most 256 axons, requires that all 256 neurons on a neurosynaptic core must receive input from a *receptive field* of the same 256 axons, and allows each neuron to transmit spikes to at most one axon line, providing

[§]Work done while at IBM Research - Almaden.

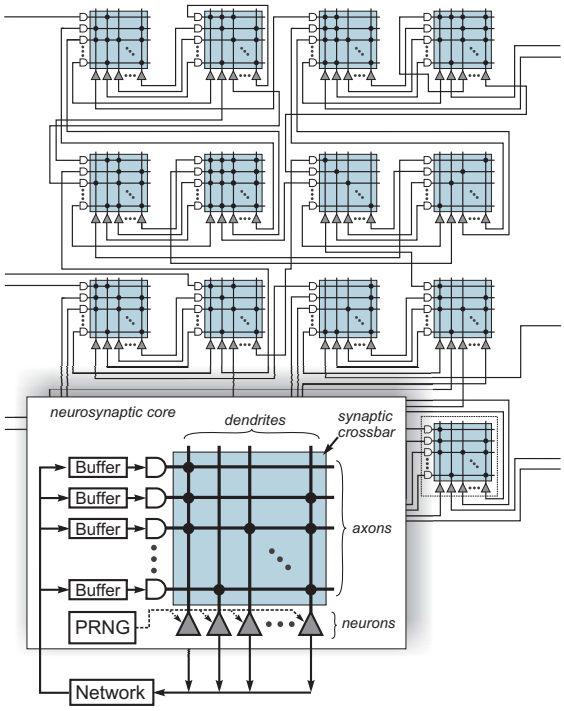


Fig. 1. TrueNorth is a brain-inspired chip architecture built from an interconnected network of lightweight neurosynaptic cores [2], [3]. TrueNorth implements "gray matter" short-range connections with an intra-core crossbar memory and "white matter" long-range connections through an inter-core spike-based message-passing network. TrueNorth is fully programmable in terms of both the "physiology" and "anatomy" of the chip, that is, neuron parameters, synaptic crossbar, and inter-core neuron-axon connectivity allow for a wide range of structures, dynamics, and behaviors. *Inset:* The TrueNorth neurosynaptic core has 256 axons, a 256×256 synapse crossbar, and 256 neurons. Information flows from axons to neurons gated by binary synapses, where each axon fans out, in parallel, to all neurons thus achieving a 256-fold reduction in communication volume compared to a point-to-point approach. A conceptual description of the core's operation follows. To support multi-valued synapses, axons are assigned types which index a synaptic weight for each neuron. Network operation is governed by a discrete time step. In a time step, if the synapse value for a particular axon-neuron pair is non-zero and the axon is active, then the neuron updates its state by the synaptic weight corresponding to the axon type. Next, each neuron applies a leak, and any neuron whose state exceeds its threshold fires a spike. Within a core, *PRNG* (pseudorandom number generator) can add noise to the spike thresholds and stochastically gate synaptic and leak updates for probabilistic computation; *Buffer* holds incoming spikes for delayed delivery; and *Network* sends spikes from neurons to axons.

a *projective field* of 256 neurons. Second, one has to optimize the implementation of each algorithm and application from the perspective of power and area by minimizing the number of neurosynaptic cores used; minimizing the average firing rate; maximizing the sparsity of synaptic crossbars; and minimizing the total time to solution. Third, one has to transduce incoming data into spike-based neural code and transduce spike-based output into the desired output format. In summary, within the hardware constraints of the TrueNorth architecture, our goal is to fully specify a network of neurosynaptic cores that carries out a desired application as efficiently as possible.

C. Contributions

Creating a TrueNorth program, a complete specification of neurosynaptic cores as well as inputs and outputs, that is consistent with the TrueNorth architecture becomes increasingly difficult as the size of a network increases. To help combat the complexity, we propose a divide-and-conquer approach whereby a large network of neurosynaptic cores is constructed by interconnecting a set of smaller networks of neurosynaptic cores, where each of the smaller networks, in turn, could be constructed by interconnecting a set of even smaller networks, and so on, until we reach a network consisting of a single neurosynaptic core, which is the fundamental, non-divisible building block.

To enable a programmer to effectively construct cognitive abstractions, algorithms, and applications that are efficient for TrueNorth, we have developed a new corelet programming paradigm that provides a complete end-to-end framework [10]. A *corelet* is an abstraction of a network of neurosynaptic cores that *encapsulates* all intra-network connectivity and all intra-core physiology and only *exposes* external inputs to and external outputs from the network. In other words, a corelet is a synonym for a program on the TrueNorth architecture. Given two or more corelets, *composition* is an operation for creating a new corelet. As algorithm designers our job is to create corelets from scratch or to compose one or more existing corelets into new corelets that are suitable for the application at hand.

Abstractions: In Section II, to carry out the essential tasks of corelet creation and composition effectively, we specify a set of repeatedly-used design abstractions that span neural codes, long-range connectivity, and short-range connectivity. In Section III, we develop and demonstrate ten algorithms and seven applications for TrueNorth.

Algorithms: The algorithms include convolution networks [11] for spatial feature extraction, spectral content estimators for time-domain to frequency-domain conversion, liquid state machines [12], [13] for feature extraction in time-varying signals, restricted Boltzmann machines (RBMs)[14] for spatial feature extraction, hidden Markov models [15] as an example of finite-state machines, looming detectors, temporal pattern matching, and various classifiers (logistic regression, backpropagation [16], stackable covariance-based). The same corelet algorithm is often used across multiple applications, and multiple corelet implementations are possible for the same algorithm, showcasing the composability and flexibility of corelet construction.

Applications: The applications include speaker recognition, music composer recognition, digit recognition, sequence prediction, collision avoidance, optical flow, and eye detection. The applications presented here were judiciously chosen to illustrate the inherent parallelism and versatility of the architecture; the architecture's ability to support feedforward, feedback, and lateral inter-core connectivity; the architecture's ability to support spatial, temporal, and spatio-temporal processing as well as event-driven sensors; the architecture's ability to support representative inputs such as voice, music, images, text, and video; the richness and diversity of intra-core connectivity; the compositional power of the programming paradigm; and the flexibility of the neuron model. All of the applications were prototyped and tested on the Compass simulator [3]. Demonstrating the scalability of TrueNorth, the systems used for these applications range in size from 38 to

3.1×10^6 neurons, 122 to 14×10^6 synaptic connections, and 1 to 21×10^3 cores.

II. Design Substrates

The representation, delivery, and integration of information in a neurosynaptic system is directly influenced by the underlying architectural substrate. Here, we describe neural codes, connection methods, and synaptic weight representation approaches used throughout the systems described in this paper.

A. Neural Codes

In a number of systems, the brain uses binary spikes to represent non-binary information [17]. Drawing inspiration from this, we use a number of neural coding schemes to represent different types of information in TrueNorth. A *binary code* uses a single spike to provide an indication that something specific has been detected. A *rate code* uses the number of spikes occurring within a specified number of time steps, the *temporal window*, to indicate the amplitude of a signal. A *population code* uses the number of spikes occurring across multiple axons or neurons, the *spatial window*, in a single time step to indicate the amplitude of a signal. *Time-to-spike coding* uses the time of a single spike in a predefined temporal window to indicate the amplitude of a signal. The particular choice of coding scheme is dependent on the neuron, axon, communication and time resources available, as well as the particular network design. We have found that rate code schemes are typically the easiest to develop systems for, but are not necessarily the most efficient in terms of resource utilization. We abbreviate coding schemes using a letter-number pair to indicate the coding scheme, Binary (B), Rate (R), Population (P), or Time-to-spike (T) followed by a number indicating the window used, such as R16 for a rate code applied in a window of 16 time steps. If the notion of window is not applicable, the number is omitted.

B. Connection Methods

The TrueNorth architecture allows each neuron to transmit spikes to a single axon line. In cases where broader connectivity is desired, a splitter corelet can be used. Each input to a splitter corelet forms a synapse with N unique neurons in that corelet, and the neurons are configured such that for each incoming spike they receive, they generate a spike of their own. Thus, a single input produces N spiking neurons on the next time step that can be targeted to N different axons.

C. Synaptic Weight Composition

The TrueNorth architecture provides the capacity for a neuron to assign four possible synaptic strengths across its inputs, with specific strength assignments dictated by the axon types at the crossbar level (see Fig. 1 for details). Further flexibility can be achieved by splitting each input and connecting it to multiple axon lines, then choosing the four available weights judiciously. For example, if each input to a core is split so that it can connect to four axon lines, each with a different type, with the corresponding strength values set to 1, 2, 4, and 8 for those types, a neuron can assign any strength between 0 and 15 to each of its inputs. A similar composition can be used if negative weights are desired. This approach can be extended to multiple axon lines if additional values are desired. Another way of emulating a large range of integer weights is to use multiple neurons and use a weighted combination of them at a later stage.

III. Applications

For each application, we describe a corelet-level connection blueprint, a description of component corelets and their associated algorithms, and results.

A. Speaker Recognition

The ability to process and integrate information from multiple sensory modalities is a hallmark of cognitive systems. Given audio and video of 34 individuals from the CUAVE database [18] speaking the digits 0–9, we constructed a system that recognizes and classifies these speakers. The system draws upon two feature extraction approaches. The first is a spectral content estimator, which computes the dot product between an input signal and a bank of square-wave filters to convert a time-varying signal into a frequency space. The second is a convolution network, which is a popular, biologically inspired approach for visual processing [11] that uses convolutions with a filter library followed by averaging to create a feature set. Features are fed into a multilayer classifier that makes successive non-linear discriminations based on linear combinations of input features. The system includes three main corelets, a spectral content estimator corelet that process audio in parallel with a convolution network corelet that process video, both of which connect to a stackable classifier corelet (Fig. 2).

Spectral Content Estimator: Input to this corelet is sent to a square-wave filter sub-corelet that implements a bank of square-wave filters with a configurable range of frequencies and phases. Each square wave is separated into positive and negative components, further divided into 256 sample segments, and a neuron is configured with a receptive field matching each segment. These neurons thus compute a portion of the total dot product of the input with a square wave. For inputs with N_S samples, this design allows representation of a square wave across $\lceil N_S/256 \rceil$ cores. In the summation corelet, a core collects all partial dot products for the same square wave. A single neuron is configured to sum the inputs, assigning a negative weight to inputs representing negative wave components and a positive weight to inputs representing positive wave components. The rate-coded output of a summation neuron thus represents the dot product of the input signal and the entire square wave.

Convolution Network: Input to this corelet is sent to a bank of convolution filter corelets, where it is divided into 16×16 patches, and each patch sent to a filter module corelet. A filter module corelet contains a single core whose neurons have receptive fields that individually represent a filter at a particular input location and collectively apply an optimal convolution of input with the desired filter. This set of corelets thereby convolves horizontal, vertical, and two diagonal edge-extraction kernels of dimension 3×3 each with the input image. Output from a convolution filter corelet is sent to an averaging corelet, containing cores configured to do an average-and-downsample operation.

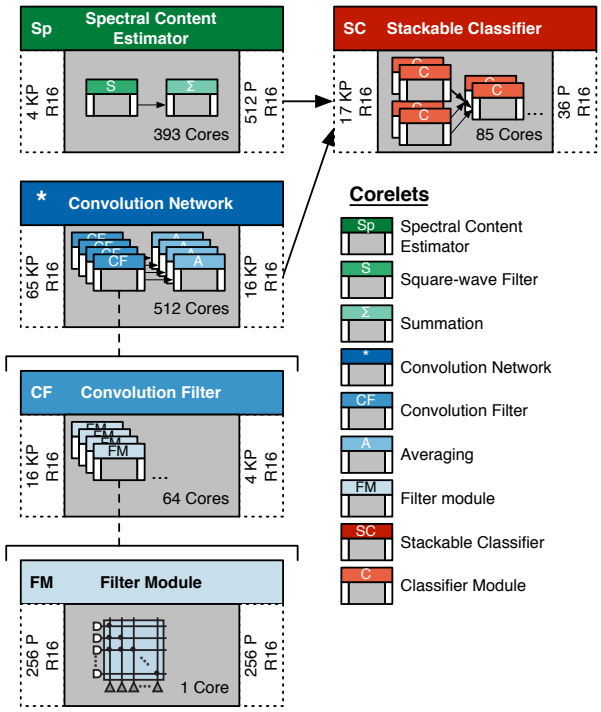


Fig. 2. Corelet diagram for the speaker recognition system. Here, and for all corelet diagrams, each box represents a corelet, with the top color bar providing the title, interior gray box showing internal structure and the side white bars representing connectors. Each connector indicates the number of pins (P), 1,024 pins (KP), or 1,048,576 of pins (MP) in the connectors, and the coding scheme used. A corelet always contains other corelets or individual cores, thus systems are always decomposable into individual cores. Input connectors link external inputs or corelet output connectors to internal corelet input connectors or axons on individual cores. Output connectors link internal corelet output connectors or neurons on individual cores to external corelet input connectors or external outputs. Here, to provide an example of the compositional structure of the system, decomposition of the convolution network corelet down to one of its single core corelets is shown.

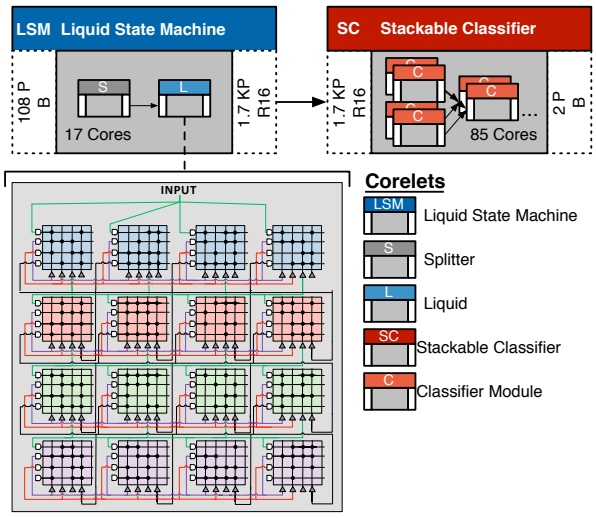


Fig. 3. Corelet diagram for the composer recognition system. The splitter sub-corelet is conditional and only generated when duplication of inputs for broader distribution is desired. A zoomed in view of the liquid corelet's core level connectivity is provided. Feedforward connections are shown in green, feedback connections are shown in black, lateral excitatory connections are shown in red and lateral inhibitory connections are shown in purple.

Stackable Classifier: This corelet uses classifier module corelets as its basic building block, where each such corelet attempts to predict the correct class label based on its own, partial set of input features. Training is accomplished by computing the covariance between a classifier module's input features and the corresponding binary class labels. Covariance values are converted into a synaptic crossbar by finding the closest fit of axon types, strength values, and binary synaptic states. By ensuring that the number of inputs to each corelet is less than or equal to 256, each corelet is implemented using a single core. For a system with N_F features and N_C classes, these corelets are arranged in a pyramid, such that $L_1 = \lceil N_F/256 \rceil$ corelets in the first layer receive external inputs. Subsequent layers are added with $L_x = \lceil (L_{x-1} * N_C)/256 \rceil$ corelets in layer L_x drawing input from the previous layer until a layer with a single corelet is reached. The rate-coded output of the top layer is read out using a winner-take-all process to provide a predicted class label.

Results: The system was trained and tested using data from 34 individuals chosen from the CUAVE database [18]. The data was processed so that each speaker's head was similarly centered and scaled in the video. Video frames were downsampled to a resolution of 75×50, converted to grayscale, and each pixel transduced using a rate code. For each video frame, the corresponding 1 second long audio segment was downsampled from 16 KHz to 4 KHz, and each audio sample transduced using rate code. 2500 total frames were used for training the system, and testing was performed on 21 out-of-sample frames from each subject. The system identified the correct speaker with 98.8% accuracy, comparable to best previous results on this dataset [19]. This system illustrates how TrueNorth's internal representation of data using the common language of spikes naturally facilitates multimodal system composition.

B. Composer Recognition

To explore the use of recurrent networks for processing time-varying signals, which are pervasive in cognitive applications, we constructed a system to distinguish between musical scores of Bach and Beethoven. Given a MIDI file dataset of musical compositions, this system uses a liquid state machine, in conjunction with the stackable classifier previously discussed, to identify the music piece's composer. Liquid state machines [12] are computational constructs that rely on recurrent connectivity to extract discriminative features from time varying signals and provide a natural and elegant approach for extracting the recurring patterns found in music. The design of the system discussed here is inspired by [20]. In Fig. 3 we provide a diagram of the system's corelet-level architecture. The system comprises three main corelets, a liquid state machine corelet that connects to an instantiation of the stackable classifier corelet described above.

Liquid State Machine: This corelet contains a splitter corelet that distributes input to a liquid corelet. The liquid corelet uses a multilayer approach, where both the number of layers and number of cores in each layer is parameterizable.

Each core allocates 27 inputs axons for each of feedforward, feedback, lateral excitatory and lateral inhibitory input, along with 27 output neurons for each of feedforward, feedback, lateral excitatory and lateral inhibitory output (see Fig. 3). Feedforward, feedback and lateral output neurons are assigned target cores uniformly in the next layer, previous layer, or same layer, respectively. Cores in the first layer receive input from the data to be processed to their feedforward input axons. All neurons in the system are duplicated (identical input weights and parameters), providing an implicit splitter, such that activity from each neuron can also be delivered to the classifier. Independent and identically distributed samples from a simulated Bernoulli random variable are used to assign the binary crossbar weights. Each of the three non-inhibitory axon types has a synaptic weight of 1, the inhibitory axon type has a weight of -1, and a neuron fires a spike when its membrane potential is at least 70.

Results: A system was created using a liquid state machine corelet with four layers and four cores per layer. The system uses the MIDI toolbox [21] to convert each music file from a dataset of 54 Bach and Beethoven MIDI files in C major to a spike-based time sequence of 108 note pitches. The music was represented using a binary code by assigning each of the 108 pitches in the music to an input line and each 32nd note to a time step. Each note produced one or more spikes on the corresponding input line and time steps. We used 44 files for training and the remainder for testing. A running total of the spikes produced by the classifier for each of the two classes (Bach or Beethoven) was computed to provide an ongoing indication of the classifier's prediction. By using winner-take-all on the final spike count per test piece, an average classification accuracy of 75% is achieved. In this system, the large number of feedback and lateral connections in the liquid state machine corelet highlights TrueNorth's suitability for recurrent connectivity.

C. Digit Recognition

Recognizing and labeling symbols from noisy sub-symbolic data is an essential capability for cognitive systems. Given handwritten digits (0–9) from the MNIST dataset [22], we developed a TrueNorth program to recognize and classify these images. This task is challenging due to variation among writing styles, even among handwriting instances from the same person repeatedly writing the same digit. To overcome this challenge, we use an RBM [14], which is a generative and unsupervised learning algorithm that clusters and extracts features from data. This ability to cluster data into similar sets without using labels is appropriate to deal with the variations in writing styles, reducing the burden on the classifier. The system comprises an RBM-trained feature corelet that extracts image features and sends them to a stochastic gradient descent trained linear classifier corelet that identifies the digit.

Coding Scheme: We implemented this network such that a new image can be input each timestep, and a recognition requires 4 timesteps to travel through the core pipeline (1 timestep for input, 1 for feature extraction, and two for classification). To achieve such throughput, we cannot use rate code, which takes many timesteps to transmit and receive a value; instead we encode all signals as either binary (spike or no spike) or as a population code. The population code we use is a thermometer code (i.e., unary code), which uses N neurons to represent values from 0 to N. If no neurons are active, the value is 0; if the first M neurons are active the value is M.

RBM Feature Extractor: This corelet comprises multiple receptive field corelets, each trained independently using an RBM (Fig. 4 *left*). Each corelet contains 64 hidden units, each excited and inhibited by a set of pixels in the image, its feedforward receptive field. A corelet is mapped to a single core by finding a best fit between learned weights and core parameters. Unlike common RBM implementations that use sigmoidal units, we threshold the hidden units' outputs at zero, resulting in a 64-dimension binary vector per receptive field corelet, which is sent to the classifier.

Linear Classifier: This corelet multiplies the binary feature extraction output with a weight matrix trained using stochastic gradient descent. We separate the classifier's weight matrix into two sublayers, simplifying the core mapping. The first sublayer uses partial sum corelets that compute the partial matrix multiplication for 64 inputs, resulting in one output per class. The second sublayer uses summation corelets to sum the partials to complete the operation. For the partial sum, the crossbars implement a 4-bit (-8..7) multiplication with the binary features. The neurons' membrane potentials correspond to the accumulation of the 16-level multiplications. We truncate the multiply-accumulate values at 24 levels, which are sent to the second sublayer, which sums them into the values for the 10 classes. The final sums are represented using an ON-OFF thermometer code, 128 output neurons corresponding to each class, representing values from -64 to 64[1]. The system codes values 1 to 64 by giving all neurons a threshold of zero and a leak ranging from -1 to -64; negative values are coded in the same manner by negating input values (in the crossbar).

Results: We preprocessed the MNIST images of handwritten digits by subsampling and thresholding, which reduced the original 28×28 pixel images down to 16×16 and reduced the number of pixel levels from 256 down to 2. This preprocessing results in a minimal loss of performance (both for machine and human), but enables us to map this task more efficiently, using fewer neurons and synapses and therefore fewer cores. We divided the image into four (overlapping) patches, corresponding to the four sets of 64 hidden units, each consisting of 11×11 pixels. We send two spikes (positive and negative axon types) for each active pixel in each patch to the hidden units. This input drives hidden unit activity in the RBM feature extractor and on to the linear classifier (Fig. 4 *center*). The system scored 92.34% correct in the 60,000 sample training set and 91.94% in the 10,000 sample test set (Fig. 4 *right*).

The digit recognition system exhibits the parallel, low-precision nature of TrueNorth computation. We use many binary feature encoder neurons in parallel (equaling the number of pixels) with a low-precision classifier, in contrast to conventional implementations, which use fewer encoder neurons and higher neuron and classifier precision to match performance.

D. HMM Sequence Modeling

Recognizing patterns within a sequence of symbols is a critical ability for cognitive systems. For example, acoustic

[1]Negative values are necessary in the case that the input image results are negative for all classes.

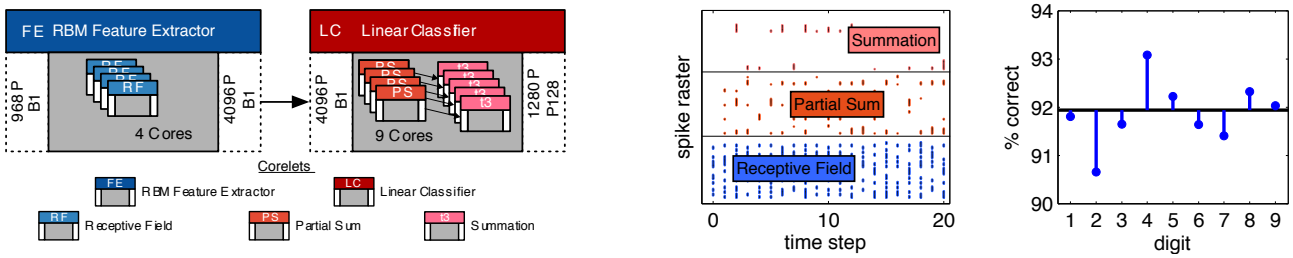


Fig. 4. *Left* The corelet diagram for the digit recognition system. *Center* The receptive field neurons sum inputs from a digit image presented at each time step and spike (dots) if excitation exceeds inhibition. Spiking receptive field neurons drive the partial sum neuron, which drive the summation neurons, the classification output (100 of each neuron type shown). *Right* The classification system achieves 91.94% correct on the 10,000 out-of-sample test digits.

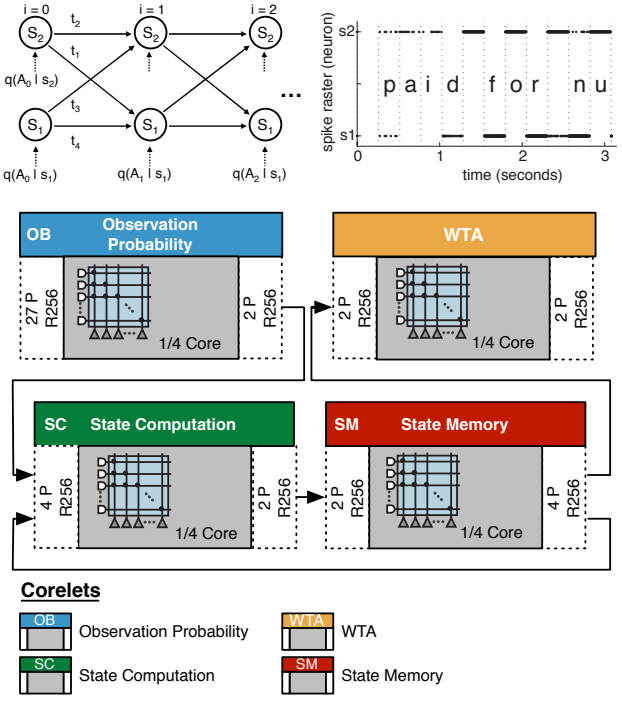


Fig. 5. Two-state HMM lattice diagram (upper left). Decoding results: s_1 corresponds to consonants and s_2 corresponds to vowels + space (upper right). Corelet diagram for the HMM sequence recognition system (bottom).

sequences form music and words, word sequences form language, and language is a construct for conscious sequential thought. Given a sample of English text, we classify the sequence of characters into two classes: consonants (s_1) or vowels and spaces (s_2) using the two-state Hidden Markov Model (HMM) [15] shown in Fig. 5 (upper left). We trained the HMM using the forward/backward algorithm on a sequence of 30,000 characters from unlabeled English text, and we mapped the resulting parameters from the HMM to a set of TrueNorth cores and tested the performance.

To evaluate the HMM, we compute the forward recursion, the probability that the HMM will end up in each state, as:

$$\begin{aligned}\alpha_i(s_1) &= \alpha_{i-1}(s_1)p(t_2)q(A_j|s_1) + \\ &\quad \alpha_{i-1}(s_2)p(t_3)q(A_j|s_1) \end{aligned} \quad (1)$$

where $\alpha_i(s_1)$ is the non-normalized probability of state 1 at time i, $p(t_x)$ is the transition probability (see Fig. 5, upper left), and $q(A_i|s_k)$ is the observation probability for the character $A \in \{a, b, c, ...\}$ at time i, for the k^{th} state. We repeat this for state 2 and map to TrueNorth corelets (Fig. 5, bottom) as follows.

Coding Scheme: Input into the HMM is represented as symbols by 27 axons, where each axon corresponds to a letter of the alphabet or the space character. A symbol is presented to the system by injecting 256 spikes—one per timestep—onto the input line corresponding to the observed character A.

Observation Probability: This corelet uses neurons with stochastic synapses to compute the observation probabilities. There is one synapse per state, per symbol. These neurons are configured with a threshold of 1 and their synapses are programmed to stochastically deliver incoming spikes with probability $q(A_i|s_k)$.

State Computation: This corelet uses three neurons per state to compute the probability $\alpha_i(s_k)$ of the system being in state s_k at time i. There are two input neurons that each compute one of the terms in (1). The system's previous state $\alpha_{i-1}(s_k)$ arrives through one axon with synaptic weight of 1, and the observation probability $q(A_i|s_k)$ arrives through another axon whose synapse is configured to stochastically deliver spikes with probability $p(t_x)$. With the threshold set at 2, each neuron outputs a spike train that probabilistically encodes $\alpha_{i-1}(s_k)p(t_x)q(A_i|s_k)$. A third neuron sums the output of the two input neurons and outputs a spike train encoding the probability $\alpha_i(s_k)$.

State Memory: This corelet stores the current and past states of the HMM using rate store neurons. These neurons are configured with a stochastic threshold and no leak so that their output spike train probabilistically encodes the value of the neuron's membrane potential. The output of these neurons is shown in Fig. 5 upper right.

WTA: Finally a winner-take-all corelet reports the maximum likelihood estimate of the class that each letter belongs to, at each time segment.

Results: We tested the performance of this HMM implemented in TrueNorth by presenting a sequence of 500 characters from out-of-sample English text (Fig. 5, upper right). Our HMM based recognizer performed with 99.8% accuracy over the 500 character test set. It is desirable to use the fewest number of neurons to implement a system on TrueNorth. We used only 8×2 neurons to compute the

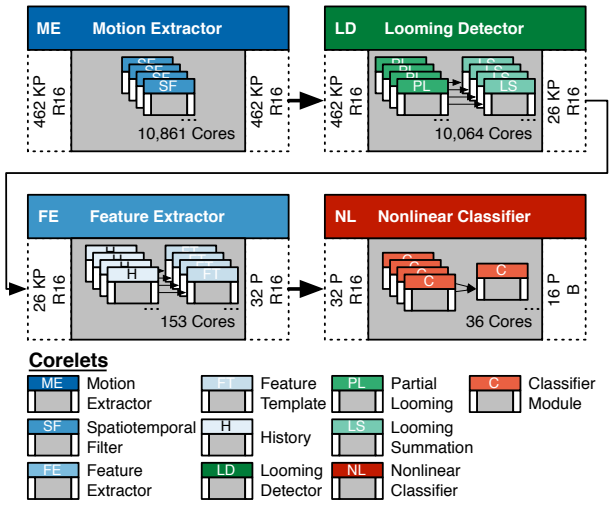


Fig. 6. Corelet diagram for the collision avoidance system.

observation probabilities, 3×2 neurons to compute the state probabilities, 2×2 neurons to store the state probabilities, 5×2 neurons for control, and 1×2 neurons for WTA (38 neurons total). In our implementation, the number of required neurons scales linearly with the number of states modeled in the HMM (with a linear scaling constant of 18 in this example.) This linear scaling generalizes to other finite-state machines implemented on the TrueNorth architecture as well.

E. Collision Avoidance

Recognizing patterns of motion is a fundamental task for a cognitive system that interprets visual signals. In particular, obstacle avoidance is essential for any navigation system. Mimicking the age-old game of dodgeball, given a series of video frames containing a ball moving towards the camera, the collision avoidance system determines whether the object would pass within one meter of the camera. Trajectories are classified into 16 target regions and a classification is ideally generated no later than 0.5 seconds before the collision. The collision avoidance system consists of four corelets: motion extractor, looming detector, feature extractor and nonlinear classifier, configured in a linear pipeline, as shown in Fig. 6.

Motion Extractor: This corelet receives transduced spikes from each input pixel, split 16 ways with appropriate delays to enable the parallel computation of the motion signal for four speeds and four directions per speed. The corelet contains spatio-temporal filter sub-corelets, each with neurons configured to receive connections from three pixels, sampled from three video frames, for a total of nine inputs per neuron. Using a spatio-temporal filtering model (as described, for example, in [23]) the summation carried out by the neuron coding the horizontal motion signal for velocity v at the pixel P located at row y, column x, for frame t is given by

$$\begin{aligned} &Mot(x,y,t,v) = \\ &\quad P(x,y,t-2) - P(x+v,y,t-2) - P(x+2v,y,t-2) - \\ &\quad P(x,y,t-1) + P(x+v,y,t-1) - P(x+2v,y,t-1) - \\ &\quad P(x,y,t) - P(x+v,y,t) + P(x+2v,y,t). \end{aligned}$$

Here $v = 1,2,3,4$ represents the velocity of the motion for rightward, and $v = -1,-2,-3,-4$ for leftward motions. A similar formula is used for vertical motion. Thus, the rate coded output of a single neuron represents the level of the motion signal for a specific speed, direction, and location.

Looming Detector: This corelet uses input from the motion extraction corelet to compute looming for each pixel—a measure of how fast the object is expanding in the field of view. This model is inspired by the locust, which has dedicated looming neurons that it uses to detect potential collisions [24], [25]. Four partial looming sub-corelets are used, each configured to compute one of the partial looming signals $R, L, U,$ or D for each pixel by using a neuron to sum the outward motion signals from two 3×5 flanking patches (Fig. 7). If this summation is positive, it is consistent with expansion of an object in the field of view; if negative, it is not consistent, and is ignored. Mathematically, this can be represented as

$$L_s(x,y) = \max(0, \sum_{j=-5}^{-1} \sum_{i=-1}^{1} l_s(x+j,y+i) - r_s(x+j,y+i)),$$

where r_s and l_s represent the rightward and leftward motion signals for speed s at a given pixel. Similar equations can be derived for the top, right and bottom windows. The final looming signal for each pixel is determined in a looming summation corelet, using neurons configured to compute

$$Loom = \sum_{s=1}^{4} R_s + L_s + U_s + D_s.$$

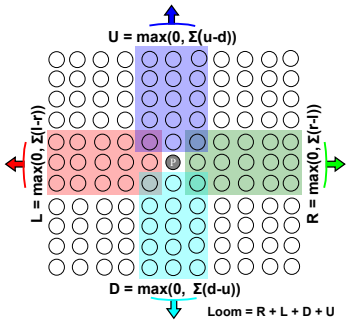


Fig. 7. 3×5 windows used to compute looming.

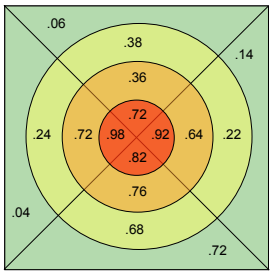


Fig. 8. The fraction of test cases classified as a hit by the collision avoidance system, in each of 16 target regions.

Feature Extractor: This corelet computes the match between recent looming activity and defined trajectory templates. A history sub-corelet uses input from the looming detection corelet to increment the membrane potential of history neurons. This membrane potential, which gradually decays over time the stochastic firing rate of these neuron. Thirty-two templates were created based on output from the history corelet by averaging a set of motion frames with similar target destinations. These were used to configure a feature template corelet such that its neurons compute an inner product between input, unknown trajectories, and known templates.

Nonlinear Classifier: This corelet assigns a predicted target location to a particular feature vector. The classifier is trained using a variant of the backpropagation algorithm [16] that we developed to converge to a TrueNorth implementable solution. Specifically, a multilayer classifier network is created following TrueNorth connectivity, then trained using standard backpropagation. After a reasonable solution is reached, the network is pushed to discrete weights by selecting a synapse at random, rounding it to the nearest integer, pinning it to that value, then further training the network for a short time (here, 50 training examples) to allow unpinned weights to improve the overall solution with this new constraint. This process is repeated until all synapses have been pinned. The resulting network is then used to configure the classifier corelet.

Results: Using USARSim[26], a virtual environment built on top of the Unreal Tournament rendering engine, video frames are generated at a 25 Hz frame rate. The image is reduced in size to 170×170 by averaging and the pixel intensities are linearly quantized to a 16-level rate code of 0–16 spikes per frame. Eight hundred videos, 50 videos of each of the 16 target areas were processed through this pipeline. Fig. 8 gives the first working results achieved.

This approach to looming detection and response demonstrates the parallel nature of TrueNorth's sub-symbolic processing. Rather than explicitly detecting and characterizing individual objects in each video frame, and carrying out sequential logical operations on such objects, this system extracts motion and looming signals in parallel across each image, and recognizes patterns in the extracted signals that are consistent with an approaching object, a method that is more flexible, robust, and scalable.

F. Optical Flow

Optical flow is the apparent motion of objects in a scene relative to an observer, often described as the direction and magnitude of motion of each pixel in a frame. In a cognitive system, optical flow can play an important role in object tracking and navigation. Given two successive frames from a video, we built a system that finds local changes in image edge locations as a measure of motion. The system is composed of four corelets: the convolution network corelet described in Section III-A, a rate to binary conversion corelet, a temporal match corelet, and an averaging corelet. Multiple instances of these corelets are instantiated and connected as illustrated in Fig. 9. For clarity, we describe here computation of optical flow along a single axis. The method described here provides an alternative to the related motion extraction corelet described in Section III-E This algorithm was inspired in functionality by [27], an algorithm developed for the "Connection Machine" at MIT in the 1980s—one of the first alternative architectures to the traditional von Neumann model of computation.

Convolution Network—Edge Extraction: This corelet is implemented as described in Section III-A, configured to extract vertical and horizontal edges in the current frame and previous frame.

Rate To Binary Conversion: In this corelet, each input line connects to a single neuron configured to provide a subthreshold summation of received spikes. A query pulse is sent to these neurons every 16 time steps, which pushes a neuron above threshold if it has received a user parameterized number of prior inputs. A reset pulse is sent in the next time step to reset these neurons. The query and reset pulses are generated by neurons configured with positive leaks to spike spontaneously at the desired frequency.

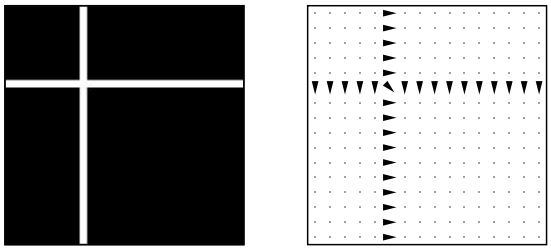


Fig. 10. For the frame shown on left, depicting movement down and to the right, computed optical flow vectors are shown on right.

Temporal Match: The output of both rate to binary conversion corelets are fed into a temporal match corelet. Axon delays are set such that input from the previous and current frame arrive simultaneously at the temporal match corelet. The temporal match corelet has neurons configured to compute an "AND" operation between a pixel in the current frame and the $\pm\ 1^{st}$ and $\pm 2^{nd}$ pixel from the delayed frame.

Convolution Network - Averaging: The temporal match corelet provides input to a bank of spatial averaging corelets. Each such corelet is a variant of the convolution network corelet, configured such that an output neuron represents an average over 3×3 pixels. The bank of averaging corelets outputs five signals for each pixel, corresponding to five velocities.

Results: Using this system, we computed optical flow using a simple test example of a horizontal bar moving downwards, and a vertical bar moving rightward (Fig. 10). This algorithm exploits the implicit parallelism of the TrueNorth architecture and illustrates a novel parallel algorithm for computing optical flow. One of the keys attribute of this algorithm is shallow, heavily parallel computation as compared to deep, sequential instruction execution on a von-Neumann architecture.

G. Eye Detection

The ability to detect and track spatio-temporal patterns is a useful feature of cognitive systems. Given an on-axis and off-axis light source, we constructed a system that uses retro-reflection to detect and track eye location, drawing inspiration from [28]. When an eye is illuminated, it reflects a narrow beam of light directly towards the light source. In flash photography this reflection causes the undesired red eye effect, but can be used advantageously for eye detection. An on-axis light induces the red-eye reflection, producing an image with bright pupils, while an off-axis light results in a similar image, but with dark pupils. Subtracting the two images provides the location of the eyes. In our approach, we use the iniLabs DVS128 spiking retina sensor [29] instead of a traditional camera. This sensor has 128×128 detectors, or pixels, each spiking asynchronously when the change in light brightness exceeds a threshold. Each pixel may spike up to 1000 times a second, and the entire sensor can deliver up to two million spikes per second. The spikes are fed into our eye detection

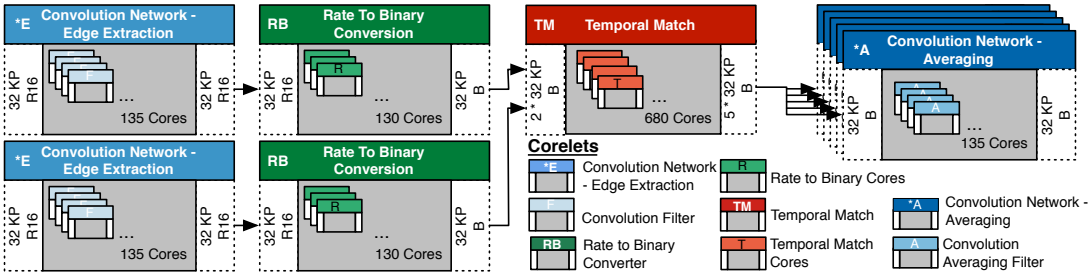


Fig. 9. Corelet diagram for the optical flow system. For clarity, corelets for optical flow along a single orientation are shown.

system, composed of a convolution and temporal integration corelet followed by an eye centroid detection corelet (Fig. 11).

Convolution and Temporal Integration: This corelet is composed of an overlap kernel tiling corelet with a 3×3 spatio-temporal kernel, implemented using the TrueNorth kernel $[a\ b\ a;\ c\ d\ c;\ a\ b\ a]$ with $a = 5, b = c = 6, d = 9$. Within this corelet, neurons are configured to compute the spatial filter, use the leak for temporal decay, and spike stochastically according to their membrane potential, like the history neurons described in Section III-E. This transform is very efficient, requiring only 1.3 neurons/pixel (average) for both the spatial and temporal filters. The filter removes much of the scattered sensory noise from the retina sensor, and integrates over time to detect pupil regions and track them. Using overlap kernel tiling, continuous coverage of the image area is achieved without causing any tile boundary effects.

Eye Centroid Detection: The eye centroid detection corelet is composed of winner-take-all corelets. Each such corelet receives an 8×8 region from the convolution and temporal integration corelet, applies a filter to find the centroid of activity in the region, and outputs a single pixel, where the maximum activity is found. This ensures that only one pixel per target is reported (Fig. 12b).

Results: Tests were conducted with simulated spike files, where targets are moved along trajectories at various directions and speeds, and with the retina camera. In this preliminary test, we use an array of three synthetic retro-reflectors as targets (5 mm in diameter). Fig. 12 shows the final result for moving targets in front of the retina camera. The use of a spiking camera for system input provides an elegant demonstration of TrueNorth's aptness for event-driven computation.

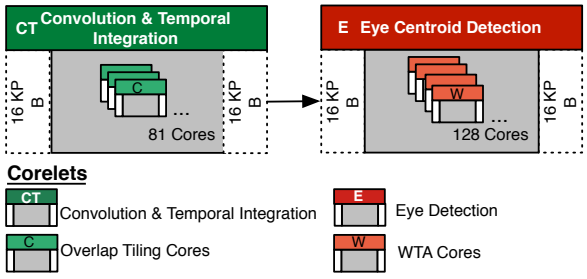
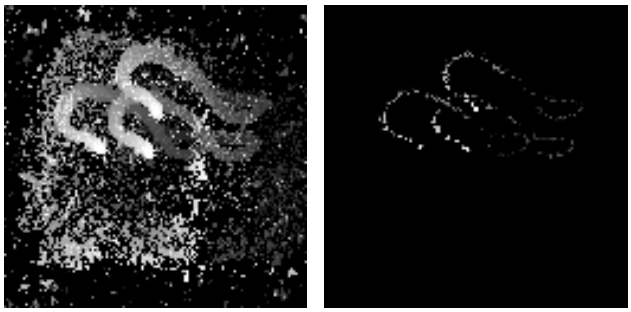


Fig. 11. Corelet diagram for the eye detection system.



(a) (b)

Fig. 12. Spikes from the retina senors (a) and the corresponding output (b), with faded history to exemplify the targets motion within a 2 seconds window.

IV. Conclusions

To realize TrueNorth's full potential as an efficient platform for running cognitive algorithms, it is necessary to go beyond the prevailing von Neumann paradigms and imagine entirely new abstractions that improve programmer productivity yet are natively efficient for neuromorphic architectures such as TrueNorth. As a step in this direction, we have leveraged a new set of programming abstractions [10], a new neuron model and libraries [9], and an architectural simulator [3] to demonstrate seven applications that are efficient on TrueNorth (in terms of the number of neurosynaptic cores used).

For the present, we have tested the applications using the Compass simulator running on a von Neumann computer—while ensuring their compatibility with forthcoming TrueNorth hardware. For the future, we are eager to measure and benchmark the power, area, and speed advantages of running our applications directly on TrueNorth hardware. Given the event-driven nature of TrueNorth, we are also interested in seeking novel event-driven sensors/actuators that seamlessly connect with it. Moving in this direction, we employed a spiking camera in our eye detection system [29], one of an emerging breed of spiking sensors [30]. Such end-to-end spike-based systems have the potential to further push the power and speed envelope while dispensing with the prevailing frame-based approaches to spatio-temporal signals.

The seven applications presented in this paper move us closer to our end goal of solving interesting and real-world problems using TrueNorth. We expect that TrueNorth's low cost (in power and size), unique architecture (scalable, parallel, distributed, event-driven), and function (multi-modal, spatio-

temporal, real-time), will open up entirely new application spaces with endless possibilities. We imagine glasses that help the visually-impaired navigate a complex environment with obstacles. We imagine cars that can understand complex scenes in real-time regardless of weather or time of day. We imagine cameras that can watch the road for accidents and alert public safety officials. We imagine sensors on patients that can anticipate and avoid emergencies. We imagine grocer's gloves that can flag bad or contaminated produce. To translate these possibilities into actualities—building on the groundwork that we have laid down here—will require a concerted collaboration of a community of researchers and practitioners who are committed to the vision of cognitive computing.

ACKNOWLEDGMENTS

This research was sponsored by DARPA under contract No. HR0011-09-C-0002. 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 DARPA or the U.S. Government. We would like to thank David Peyton for his expert assistance revising this manuscript.

REFERENCES

[1] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. J. Sherbondy, and R. Singh, "Cognitive computing," *Communications of the ACM*, vol. 54, no. 8, pp. 62–71, 2011.

[2] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," in *IEEE Custom Integrated Circuits Conference (CICC)*, Sept. 2011, pp. 1–4.

[3] 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 for High Performance Computing, Networking, Storage, and Analysis (SC 2012)*, Nov. 2012, p. 54.

[4] J. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman, "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *IEEE Custom Integrated Circuits Conference (CICC)*, Sept. 2011, pp. 1–4.

[5] J. V. Arthur, P. A. Merolla, F. Akopyan, R. Alvarez, A. S. Cassidy, S. Chandra, S. K. Esser, N. Imam, W. Risk, D. B. D. Rubin, R. Manohar, and D. S. Modha, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *The International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1–8.

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

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

[8] E. McQuinn, P. Datta, M. D. Flickner, W. P. Risk, D. S. Modha, T. M. Wong, R. Singh, S. K. Esser, and R. Appuswamy, "2012 international science & engineering visualization challenge," *Science*, vol. 339, no. 6119, pp. 512–513, February 2013.

[9] A. S. Cassidy, P. Merolla, J. V. Arthur, S. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, A. Amir, D. Rubin, F. Akopyan, E. McQuinn, W. Risk, and D. S. Modha, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013.

[10] A. Amir, P. Datta, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, W. Risk, M. Flickner, R. Alvarez-Icaza, E. McQuinn, B. Shaw, N. Pass, and D. S. Modha, "Cognitve computing programming paradigm: A corelet language for composing networks of neuro-synaptic cores," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013.

[11] Y. Le Cun, J. Boser, D. Denker, R. Henderson, W. Howard, W. Hubbard, and L. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 4, no. 1, pp. 541–551, 1989.

[12] W. Maass, "Liquid state machines: Motivation, theory, and applications," in *Computability in Context: Computation and Logic in the Real World*, B. Cooper and A. Sorbi, Eds. Imperial College Press, 2010, pp. 275–296.

[13] W. Maass, N. T., and M. H., "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[14] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. [Online]. Available: http://dx.doi.org/10.1126/science.1127647

[15] F. Jelinek, *Statistical methods for speech recognition*. MIT press, 1998.

[16] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[17] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, Cambridge, 2002.

[18] E. Patterson, S. Gurbuz, Z. Tufekci, and J. Gowdy, "Cuave: A new audio-visual database for multimodal human-computer interface research," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2. IEEE, 2002, pp. 2017–2020.

[19] D. Dean, P. Lucey, and S. Sridharan, "Audio-visual speaker identification using the cuave database," in *Proceedings auditory-visual speech processing 2005, AVSP05*, 2005.

[20] L. Pape, J. de Gruijl, and M. Wiering, *Speech, Audio, Image and Biomedical Signal Processing Using Neural Networks*. Springer, 2008, ch. Democratic Liquid State Machines for Music Recognition, pp. 191–215.

[21] T. Eerola and P. Toiviainen. (2004) Midi toolbox: Matlab tools for music research. University of Jyväskylä, Finland.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *J. OPT. SOC. AM. A*, vol. 2, no. 2, pp. 284–299, 1985.

[24] S. Yue and F. C. Rind, "Collision detection in complex dynamic scenes using an LGMD-based visual neural network with feature enhancement," *Trans. Neur. Netw.*, vol. 17, no. 3, pp. 705–716, May 2006. [Online]. Available: http://dx.doi.org/10.1109/TNN.2006.873286

[25] S. Bermudez i Badia and P. Verschure, "A collision avoidance model based on the lobula giant movement detector (LGMD) neuron of the locust," in *IEEE International Joint Conference on Neural Networks*, vol. 3, july 2004, pp. 1757 – 1761 vol.3.

[26] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: a robot simulator for research and education," in *2007 IEEE International Conference on Robotics and Automation*, april 2007, pp. 1400–1405.

[27] H. Bülthoff, J. Little, and T. Poggio, "A parallel algorithm for real-time computation of optical flow," *Nature*, vol. 337, no. 6207, pp. 549–553, 1989.

[28] C. Morimoto, D. Koons, A. Amir, and M. Flickner, "Pupil detection and tracking using multiple light sources," *Image and Vision Computing*, vol. 18, no. 4, pp. 331–335, March 2000.

[29] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$ 128 120 db 30 mw asynchronous vision sensor that responds to relative intensity change," *IEEE ISSCC Digest of Technical Papers*, pp. 2060–2069, feb. 2006.

[30] S. Liu, A. van Schaik, B. Minch, and T. Delbruck, "Event-based 64-channel binaural silicon cochlea with q enhancement mechanisms," *ISCAS*, pp. 2027–2030, 2010.