# Convolutional networks for fast, energy-efficient neuromorphic computing

Steven K. Esser[a,1], Paul A. Merolla[a], John V. Arthur[a], Andrew S. Cassidy[a], Rathinakumar Appuswamy[a], Alexander Andreopoulos[a], David J. Berg[a], Jeffrey L. McKinstry[a], Timothy Melano[a], Davis R. Barch[a], Carmelo di Nolfo[a], Pallab Datta[a], Arnon Amir[a], Brian Taba[a], Myron D. Flickner[a], and Dharmendra S. Modha[a]

[a]Brain-Inspired Computing, IBM Research–Almaden, San Jose, CA 95120

Edited by Karlheinz Meier, University of Heidelberg, Heidelberg, Germany, and accepted by Editorial Board Member J. A. Movshon August 9, 2016 (received for review March 24, 2016)

**Deep networks are now able to achieve human-level performance on a broad spectrum of recognition tasks. Independently, neuromorphic computing has now demonstrated unprecedented energy-efficiency through a new chip architecture based on spiking neurons, low precision synapses, and a scalable communication network. Here, we demonstrate that neuromorphic computing, despite its novel architectural primitives, can implement deep convolution networks that (*i*) approach state-of-the-art classification accuracy across eight standard datasets encompassing vision and speech, (*ii*) perform inference while preserving the hardware's underlying energy-efficiency and high throughput, running on the aforementioned datasets at between 1,200 and 2,600 frames/s and using between 25 and 275 mW (effectively >6,000 frames/s per Watt), and (*iii*) can be specified and trained using backpropagation with the same ease-of-use as contemporary deep learning. This approach allows the algorithmic power of deep learning to be merged with the efficiency of neuromorphic processors, bringing the promise of embedded, intelligent, brain-inspired computing one step closer.**

convolutional network | neuromorphic | neural network | TrueNorth

The human brain is capable of remarkable acts of perception while consuming very little energy. The dream of brain-inspired computing is to build machines that do the same, requiring high-accuracy algorithms and efficient hardware to run those algorithms. On the algorithm front, building on classic work on backpropagation (1), the neocognitron (2), and convolutional networks (3), deep learning has made great strides in achieving human-level performance on a wide range of recognition tasks (4). On the hardware front, building on foundational work on silicon neural systems (5), neuromorphic computing, using novel architectural primitives, has recently demonstrated hardware capable of running 1 million neurons and 256 million synapses for extremely low power (just 70 mW at real-time operation) (6). Bringing these approaches together holds the promise of a new generation of embedded, real-time systems, but first requires reconciling key differences in the structure and operation between contemporary algorithms and hardware. Here, we introduce and demonstrate an approach we call Eedn, energy-efficient deep neuromorphic networks, which creates convolutional networks whose connections, neurons, and weights have been adapted to run inference tasks on neuromorphic hardware.

For structure, typical convolutional networks place no constraints on filter sizes, whereas neuromorphic systems can take advantage of blockwise connectivity that limits filter sizes, thereby saving energy because weights can now be stored in local on-chip memory within dedicated neural cores. Here, we present a convolutional network structure that naturally maps to the efficient connection primitives used in contemporary neuromorphic systems. We enforce this connectivity constraint by partitioning filters into multiple groups and yet maintain network integration by interspersing layers whose filter support region is able to cover incoming features from many groups by using a small topographic size (7).

For operation, contemporary convolutional networks typically use high precision (≥32-bit) neurons and synapses to provide continuous derivatives and support small incremental changes to network state, both formally required for backpropagation-based gradient learning. In comparison, neuromorphic designs can use one-bit spikes to provide event-based computation and communication (consuming energy only when necessary) and can use low-precision synapses to colocate memory with computation (keeping data movement local and avoiding off-chip memory bottlenecks). Here, we demonstrate that by introducing two constraints into the learning rule—binary-valued neurons with approximate derivatives and trinary-valued ({−1,0,1}) synapses—it is possible to adapt backpropagation to create networks directly implementable using energy efficient neuromorphic dynamics. This approach draws inspiration from the spiking neurons and low-precision synapses of the brain (8) and builds on work showing that deep learning can create networks with constrained connectivity (9), low-precision synapses (10, 11), low-precision neurons (12–14), or both low-precision synapses and neurons (15, 16). For input data, we use a first layer to transform multivalued, multichannel input into binary channels using convolution filters that are learned via backpropagation (12, 16) and whose output can be sent on chip in the form of spikes. These binary channels, intuitively akin to independent

## Significance

Brain-inspired computing seeks to develop new technologies that solve real-world problems while remaining grounded in the physical requirements of energy, speed, and size. Meeting these challenges requires high-performing algorithms that are capable of running on efficient hardware. Here, we adapt deep convolutional neural networks, which are today's state-of-the-art approach for machine perception in many domains, to perform classification tasks on neuromorphic hardware, which is today's most efficient platform for running neural networks. Using our approach, we demonstrate near state-of-the-art accuracy on eight datasets, while running at between 1,200 and 2,600 frames/s and using between 25 and 275 mW.

Author contributions: S.K.E., P.A.M., J.V.A., R.A., and D.S.M. designed research; S.K.E., P.A.M., J.V.A., A.S.C., R.A., A. Andreopoulos, D.J.B., J.L.M., T.M., D.R.B., C.d.N., P.D., A. Amir, B.T., and M.D.F. performed research; S.K.E. contributed new reagents/analytic tools; S.K.E., P.A.M., J.V.A., A.S.C., R.A., and A. Andreopoulos analyzed data; and S.K.E., P.A.M., A.S.C., and D.S.M. wrote the paper.

Conflict of interest statement: All authors are employees of IBM Research.

This article is a PNAS Direct Submission. K.M. is a Guest Editor invited by the Editorial Board.

Freely available online through the PNAS open access option.

See Commentary on page 11387.

[1]To whom correspondence should be addressed. Email: sesser@us.ibm.com.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1604850113/-/DCSupplemental.

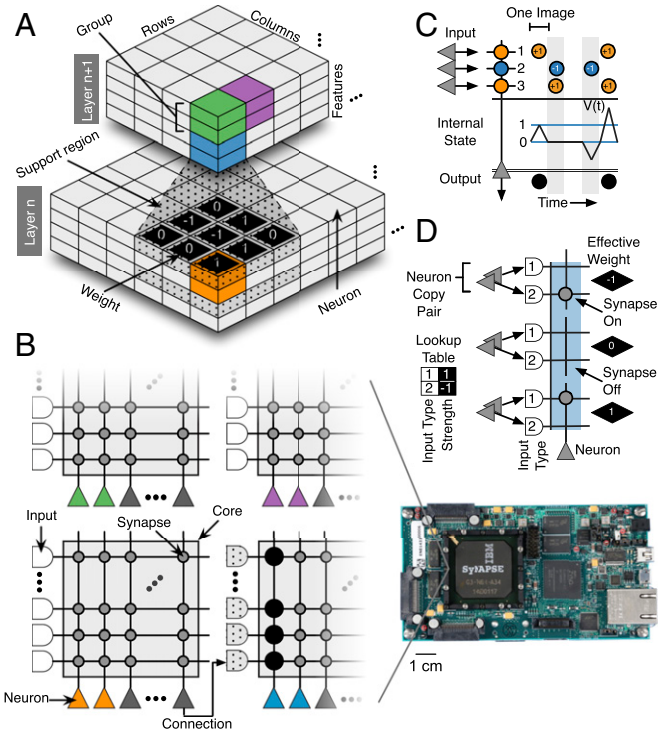


**Fig. 1.** (*A*) Two layers of a convolutional network. Colors (green, purple, blue, orange) designate neurons (individual boxes) belonging to the same group (partitioning the feature dimension) at the same location (partitioning the spatial dimensions). (*B*) A TrueNorth chip (shown far right socketed in IBM's NS1e board) comprises 4,096 cores, each with 256 inputs, 256 neurons, and a 256 × 256 synaptic array. Convolutional network neurons for one group at one topographic location are implemented using neurons on the same TrueNorth core (TrueNorth neuron colors correspond to convolutional network neuron colors in *A*), with their corresponding filter support region implemented using the core's inputs, and filter weights implemented using the core's synaptic array. (*C*) Neuron dynamics showing that the internal state variable $V(t)$ of a TrueNorth neuron changes in response to positive and negative weighted inputs. Following input integration in each tick, a spike is emitted if $V(t)$ is greater than or equal to the threshold $\theta = 1$. $V(t)$ is reset to 0 before input integration in the next tick. (*D*) Convolutional network filter weights (numbers in black diamonds) implemented using TrueNorth, which supports weights with individually configured on/off state and strength assigned by lookup table. In our scheme, each feature is represented with pairs of neuron copies. Each pair connects to two inputs on the same target core, with the inputs assigned types 1 and 2, which via the look up table assign strengths of +1 or −1 to synapses on the corresponding input lines. By turning on the appropriate synapses, each synapse pair can be used to represent −1, 0, or +1.

components (17) learned with supervision, provide a parallel distributed representation to carry out high-fidelity computation without the need for high-precision representation.

**Table 1. Structure of convolution networks used in this work**

| 1/2 chip | 1 chip | 2 chip | 4 chip |
|---|---|---|---|
| S-12 | S-16 | S-32 | S-64 |
| P4-128 (4) | P4-252 (2) | S-128 (4) | S-256 (8) |
| D | N-256 (2) | N-128 (1) | N-256 (2) |
| S-256 (16) | P-256 (8) | P-128 (4) | P-256 (8) |
| N-256 (2) | S-512 (32) | S-256 (16) | S-512 (32) |
| P-512 (16) | N-512 (4) | N-256 (2) | N-512 (4) |
| S-1020 (4) | N-512 (4) | P-256 (8) | P-512 (16) |
| (6,528/class) | N-512 (4) | S-512 (32) | S-1024 (64) |
| | P-512 (16) | N-512 (4) | N-1024 (8) |
| | S-1024 (64) | P-512 (16) | P-1024 (32) |
| | N-1024 (8) | S-2048 (64) | S-2048 (128) |
| | P-1024 (32) | N-2048 (16) | N-2048 (16) |
| | N-1024 (8) | N-2048 (16) | N-2048 (16) |
| | N-1024 (8) | N-2048 (16) | N-2048 (16) |
| | N-2040 (8) | N-4096 (16) | N-4096 (16) |
| | (816/class) | (6,553/class) | (6,553/class) |

Each layer is described as type-features (groups), where type can be S for spatial filter layers with filter size 3 × 3 and stride 1, N for network-in-network layers with filter size 1 × 1 and stride 1, P for convolutional pooling layer with filter size 2 × 2 and stride 2, P4 for convolutional pooling layer with filter size 4 × 4 and stride 2, and D for dropout layers. The number of output features assigned to each of the 10 CIFAR10 classes is indicated below the final layer as (features/class). The eight-chip network is the same as a four-chip network with twice as many features per layer.

Critically, we demonstrate that bringing the above innovations together allows us to create networks that approach state-of-the-art accuracy performing inference on eight standard datasets, running on a neuromorphic chip at between 1,200 and 2,600 frames/s (FPS), using between 25 and 275 mW. We further explore how our approach scales by simulating multichip configurations. Ease-of-use is achieved using training tools built from existing, optimized deep learning frameworks (18), with learned parameters mapped to hardware using a high-level deployment language (19). Although we choose the IBM TrueNorth chip (6) for our example deployment platform, the essence of our constructions can apply to other emerging neuromorphic approaches (20–23) and may lead to new architectures that incorporate deep learning and efficient hardware primitives from the ground up.

## Approach

Here, we provide a description of the relevant elements of deep convolutional networks and the TrueNorth neuromorphic chip and describe how the essence of the former can be realized on the latter.

**Deep Convolutional Networks.** A deep convolutional network is a multilayer feedforward neural network, whose input is typically image-like and whose layers are neurons that collectively

**Table 2. Summary of datasets**

| Dataset | Classes | Input | Description |
|---|---|---|---|
| CIFAR10 (26) | 10 | 32 row × 32 column × 3 RGB | Natural and manufactured objects in their environment |
| CIFAR100 (26) | 100 | 32 row × 32 column × 3 RGB | Natural and manufactured objects in their environment |
| SVHN (27) | 10 | 32 row × 32 column × 3 RGB | Single digits of house addresses from Google's Street View |
| GTSRB (28) | 43 | 32 row × 32 column × 3 RGB | German traffic signs in multiple environments |
| Flickr-Logos32 (29) | 32 | 32 row × 32 column × 3 RGB | Localized corporate logos in their environment |
| VAD (30, 31) | 2 | 16 sample × 26 MFCC | Voice activity present or absent, with noise (TIMIT + NOISEX) |
| TIMIT class (30). | 39 | 32 sample × 16 MFCC × 3 delta | Phonemes from English speakers, with phoneme boundaries |
| TIMIT frame (30) | 39 | 16 sample × 39 MFCC | Phonemes from English speakers, without phoneme boundaries |

GTSRB and Flickr-Logos32 are cropped and/or downsampled from larger images. VAD and TIMIT datasets have Mel-frequency cepstral coefficients (MFCC) computed from 16-kHz audio data.

perform a convolutional filtering of the input or a prior layer (Fig. 1*A*). Neurons within a layer are arranged in two spatial dimensions, corresponding to shifts in the convolution filter, and one feature dimension, corresponding to different filters. Each neuron computes a summed weighted input, $s$, as

$$s = \sum_{i,j} \sum_{f} x_{i,j,f} w_{i,j,f}, \quad [1]$$

where $\mathbf{x} = \{x_{i,j,f}\}$ are the neuron's input pixels or neurons, $\mathbf{w} = \{w_{i,j,f}\}$ are the filter weights, $i$, $j$ are over the topographic dimensions, and $f$ is over the feature dimension or input channels. Batch normalization (24) can be used to zero center $s$ and normalize its standard deviation to 1, following

$$r = \frac{s - \mu}{\sigma + \epsilon} + b, \quad [2]$$

where $r$ is the filter response, $b$ is a bias term, $\epsilon = 10^{-4}$ provides numerical stability, and $\mu$ and $\sigma$ are the mean and standard deviation of $s$ computed per filter using all topographic locations and examples in a data batch during training, or using the entire training set during inference. Final neuron output is computed by applying a nonlinear activation function to the filter response, typically a rectified linear unit that sets negative values to 0 (25). In a common scheme, features in the last layer are each assigned a label—such as prediction class—and vote to formulate network output (7).

Deep networks are trained using the backpropagation learning rule (1). This procedure involves iteratively (*i*) computing the network's response to a batch of training examples in a forward pass, (*ii*) computing the error between the network's output and the desired output, (*iii*) using the chain rule to compute the error gradient at each synapse in a backward pass, and (*iv*) making a small change to each weight along this gradient so as to reduce error.

**TrueNorth.** A TrueNorth chip consists of a network of neurosynaptic cores with programmable connectivity, synapses, and neuron parameters (Fig. 1*B*). Connectivity between neurons follows a blockwise scheme: each neuron can connect to one input line of any core in the system, and from there to any neuron on that core through local synapses. All communication to-, from-, and within-chip is performed using spikes.

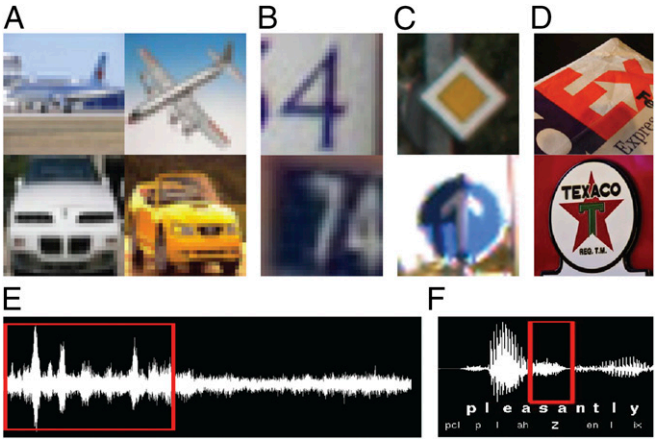


**Fig. 2.** Dataset samples. (*A*) CIFAR10 examples of airplane and automobile. (*B*) SVHN examples of the digits 4 and 7. (*C*) GTSRB examples of the German traffic signs for priority road and ahead only. (*D*) Flickr-Logos32 examples of corporate logos for FedEx and Texaco. (*E*) VAD example showing voice activity (red box) and no voice activity at 0 dB SNR. (*F*) TIMIT examples of the phonemes pcl, p, l, ah, z (red box), en, l, and ix.

**Fig. 3.** Each row shows an example image from CIFAR10 (column 1) and the corresponding output of 12 typical transduction filters (columns 2–13).

TrueNorth neurons use a variant of an integrate-and-fire model with 23 configurable parameters where a neuron's state variable, $V(t)$, updates each tick, $t$—typically at 1,000 ticks/s, although higher rates are possible—according to

$$V(t+1) = V(t) + \sum_{i} \hat{x}_i(t) w_i + L, \quad [3]$$

where $\hat{\boldsymbol{x}}(t) = \{\hat{x}_i\}$ are the neuron's spiking inputs, $\mathbf{w} = \{w_i\}$ are its corresponding weights, $L$ is its leak chosen from $\{-255, -254, \ldots, 255\}$, and $i$ is over its inputs. If $V(t)$ is greater than or equal to a threshold $\theta$, the neuron emits a spike and resets using one of several reset modes, including resetting to 0. If $V(t)$ is below a lower bound, it can be configured to snap to that bound.

Synapses have individually configurable on/off states and have a strength assigned by look-up table. Specifically, each neuron has a four-entry table parameterized with values in the range $\{-255, -254, \ldots, 255\}$, each input line to a core is assigned an input type of 1, 2, 3, or 4, and each synapse then determines its strength by using the input type on its source side to index into the table of the neuron on its target side.* In this work, we only use two input types, corresponding to synapse strengths of −1 and 1, described in the next section.

**Mapping Deep Convolutional Networks to TrueNorth.** By appropriately designing the structure, neurons, network input, and weights of convolutional networks during training, it is possible to efficiently map those networks to neuromorphic hardware.

***Structure.*** Network structure is mapped by partitioning each layer into 1 or more equally sized groups along the feature dimension,† where each group applies its filters to a different, nonoverlapping, equally sized subset of layer input features. Layers are designed such that the total filter size (rows × columns × features) of each group is less than or equal to the number of input lines available per core, and the number of output features is less than or equal to the number of neurons per core. This arrangement allows one group's features, filters, and filter support region to be implemented using one core's neurons, synapses, and input lines, respectively (Fig. 1*B*). Total filter size was further limited to 128 here, to support trinary synapses, described below. For efficiency, multiple topographic locations for the same group can be implemented on the same core. For example, by delivering a $4 \times 4 \times 8$ region of the input space to a single core, that core can be used to implement overlapping filters of size $3 \times 3 \times 8$ for four topographic locations.

Where filters implemented on different cores are applied to overlapping regions of the input space, the corresponding input neurons must target multiple cores, which is not explicitly supported by TrueNorth. In such instances, multiple neurons on the

*It should be noted that our approach can easily be adapted to hardware with other synaptic representation schemes.

†Feature groups were originally used by AlexNet (25), which split the network to run on two parallel GPUs during training. The use of grouping is expanded upon considerably in this work.

**Table 3. Summary of results**

| | State of the art | | TrueNorth best accuracy | | TrueNorth 1 chip | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Dataset | Approach | Accuracy | Accuracy | #cores | Accuracy | #cores | FPS | mW | FPS/W |
| CIFAR10 | CNN (11) | 91.73% | **89.32%** | 31492 | **83.41%** | 4042 | 1249 | 204.4 | 6108.6 |
| CIFAR100 | CNN (34) | 65.43% | **65.48%** | 31492 | **55.64%** | 4042 | 1526 | 207.8 | 7343.7 |
| SVHN | CNN (34) | 98.08% | **97.46%** | 31492 | **96.66%** | 4042 | 2526 | 256.5 | 9849.9 |
| GTSRB | CNN (35) | 99.46% | **97.21%** | 31492 | **96.50%** | 4042 | 1615 | 200.6 | 8051.8 |
| LOGO32 | CNN | 93.70% | **90.39%** | 13606 | **85.70%** | 3236 | 1775 | 171.7 | 10335.5 |
| VAD | MLP (36) | 95.00% | **97.00%** | 1758 | **95.42%** | 423 | 1539 | 26.1 | 59010.7 |
| TIMIT Class. | HGMM (37) | 83.30% | **82.18%** | 8802 | **79.16%** | 1943 | 2610 | 142.6 | 18300.1 |
| TIMIT Frames | BLSTM (38) | 72.10% | **73.46%** | 20038 | **71.17%** | 2476 | 2107 | 165.9 | 12698.0 |

The network for LOGO32 was an internal implementation. BLSTM, bidirectional long short-term memory; CNN, convolutional neural network; FPS, frames/second; FPS/W, fames/second per Watt; HGMM, hierarchical Gaussian mixture model; MLP, multilayer perceptron. Accuracy of TrueNorth networks is shown in bold.

same core are configured with identical synapses and parameters (and thus will have matching output), allowing distribution of the same data to multiple targets. If insufficient neurons are available on the same core, a feature can be "split" by connecting it to a core with multiple neurons configured to spike whenever they receive a spike from that feature. Neurons used in either duplication scheme are referred to here as copies.

***Neurons.*** To match the use of spikes in hardware, we use a binary representation scheme for data throughout the network.[‡] Neurons in the convolutional network use the activation function

$$y = \begin{cases} 1 & \text{if } r \geq 0, \\ 0 & \text{otherwise,} \end{cases} \quad [4]$$

where $y$ is neuron output and $r$ is the neuron filter response (Eq. **2**). By configuring TrueNorth neurons such that (*i*) $L = \lceil b(\sigma + \epsilon) - \mu \rceil$, where $L$ is the leak from Eq. **3** and the remaining variables are the normalization terms from Eq. **2**, which are computed from training data offline, (*ii*) threshold ($\theta$ in Eq. **2**) is 1, (*iii*) reset is to 0 after spiking, and (*iv*) the lower bound on the membrane potential is 0, their behavior exactly matches that in Eq. **3** (Fig. 1*C*). Conditions *iii* and *iv* ensure that $V(t)$ is 0 at the beginning of each image presentation, allowing for one classification per tick using pipelining.

***Network input.*** Network inputs are typically represented with multibit channels [for example, eight-bit red, green, and blue (RGB) channels]. Directly converting the state of each bit into a spike would result in an unnatural neural encoding because each bit represents a different value (for example, the most-significant-bit spike would carry a weight of 128 in an eight-bit scheme). Here, we avoid this awkward encoding altogether by converting the high precision input into a spiking representation using convolution filters with the binary output activation function described in Eq. **4**. This process is akin to the transduction that takes place in biological sensory organs, such as the conversion of brightness levels into single spikes representing spatial luminance gradients in the retina.

***Weights.*** Although TrueNorth does not directly support trinary weights, they can be simulated by using neuron copies such that a feature's output is delivered in pairs to its target cores. One member of the pair is assigned input type 1, which corresponds to a +1 in every neuron's lookup table, and the second input type 2, which corresponds to a −1. By turning on neither, one, or the other of the corresponding synaptic connections, a weight of 0, +1, or −1 can be created (Fig. 1*D*). To allow us to map into this representation, we restrict synaptic weights in the convolutional network to these same trinary values.

***Training.*** Training was performed using standard backpropagation with batch normalization (24), incorporating the activation function and constraints on receptive field sizes using groups described above, and using an approximate neuron derivative, weight update with hysteresis, and spike sparsity pressure. Details of network initialization and training are provided in the *SI Appendix*, Algorithms 1 and 2.

As the binary-valued neuron used here has a derivative of ∞ at 0, and 0 everywhere else, which is not amenable to backpropagation, we instead approximate its derivative as being 1 at 0 and linearly decaying to 0 in the positive and negative direction according to

$$\frac{\partial y}{\partial r} \approx \max(0, 1 - |r|), \quad [5]$$

where $r$ is the filter response, and $y$ is the neuron output. Weight updates are applied to a high precision hidden value, $w^h$, which is bounded in the range −1 to 1 by clipping, and mapped to the trinary value used for the forward and backward pass by rounding with hysteresis according to

$$w(t) = \begin{cases} -1 & \text{if } w^h(t) \leq -0.5 - h, \\ 0 & \text{if } w^h(t) \geq -0.5 + h \vee w^h(t) \leq 0.5 - h, \\ 1 & \text{if } w^h(t) \geq 0.5 + h, \\ w(t-1) & \text{otherwise,} \end{cases} \quad [6]$$

where $h$ is a hysteresis parameter set to 0.1 here.[§] Hysteresis prevents weights from rapidly oscillating between integer values if the corresponding hidden weight is near −0.5 or 0.5. The hidden weights allows synapses to flip between discrete states based on subtle differences in the relative amplitude of error gradients measured across multiple training batches.

We use standard heuristics for training, including momentum (0.9), weight decay ($10^{-7}$), and decreasing learning rate (dropping by 10× twice during training). We further use a spike sparsity pressure by adding $\gamma \frac{1}{2} \sum \bar{y}^2$ to the cost function, where $\bar{y}$ is average feature activation, the summation is over all features in the network, and γ is a parameter, set to $10^{-4}$ here. The sparsity pressure serves as a regularizer and to reduce spike traffic (and thus energy consumption) during deployment.

Training was performed offline on conventional GPUs, using a library of custom training layers built on functions from the MatConvNet toolbox (18). Network specification and training complexity using these layers is on par with standard deep learning.

[‡]Schemes that use higher precision are possible, such as using the number of spikes generated in a given time window to represent data (a rate code). However, we observed the best accuracy for a given energy budget by using the binary scheme described here.

[§]This rule is similar to the recent results from BinaryNet (16), but was developed independently here in this work. Our specific neuron derivative and use of hysteresis are unique.

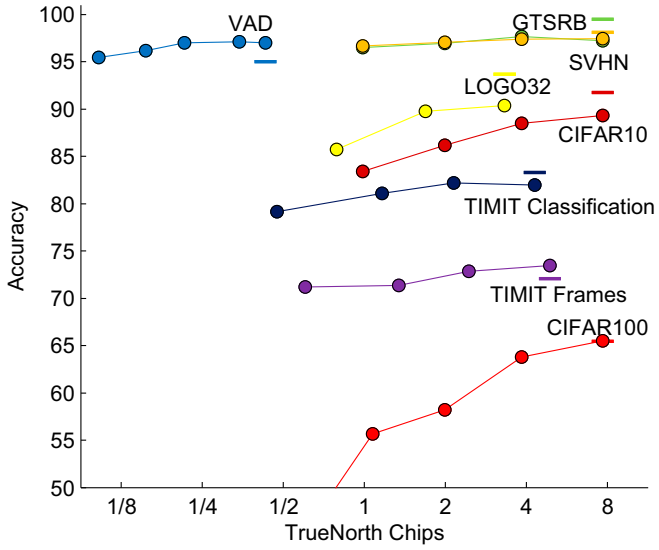


**Fig. 4.** Accuracy of different sized networks running on one or more TrueNorth chips to perform inference on eight datasets. For comparison, accuracy of state-of-the-art unconstrained approaches are shown as bold horizontal lines (hardware resources used for these networks are not indicated).

***Deployment.*** The parameters learned through training are mapped to hardware using reusable, composable hardware description functions called corelets (19). The corelets created for this work automatically compile the learned network parameters, which are independent of any neuromorphic platform, into a platform-specific hardware configuration file that can directly program TrueNorth chips.

## Results

We applied our approach to eight image and audio benchmarks using five network structures that require 0.5, 1, 2, 4, or 8 TrueNorth chips[¶] (Tables 1 and 2 and Fig. 2). Testing was performed at one classification per hardware tick.

**Networks.** Three layer configurations were especially useful in this work, although our approach supports a variety of other parameterizations. First, spatial filter layers use patch size $3\times3\times8$ and stride 1, allowing placement of four topographic locations per core. Second, network-in-network layers (7) use patch size $1\times1\times128$ and stride of 1, allowing each filter to span a large portion of the incoming feature space, thereby helping to maintain network integration. Finally, pooling layers use standard convolution layers (32) with patch size $2\times2\times32$ and stride 2, thereby resulting in nonoverlapping patches that reduce the need for neuron copies.

We found that using up to 16 channels for the transduction layer (Fig. 3) gave good performance at a low bandwidth. For multichip networks, we used additional channels, presupposing additional bandwidth in larger systems. As smaller networks required less regularization, weight decay was not used for networks smaller than four chips, and spike sparsity pressure was not used for networks half chip size or less.

**Hardware.** To characterize performance, all networks that fit on a single chip were run in TrueNorth hardware. Multichip networks require tools for efficient core placement to maximize the fraction of traffic routed on-chip rather than between chips, as intrachip bandwidth is higher than interchip bandwidth. As such tools are presently undergoing development, we chose to run multichip networks in simulation (33). In all cases, the first convolutional layer (the transduction layer) was computed off chip, in the process converting the multivalued input into a binary representation. The corresponding data were then delivered to the chip using spike packets sent over a custom asynchronous link (6). Single-chip classification accuracy and throughput were measured on the NS1e development board (Fig. 1*B*), but power was measured on a separate NS1t test and characterization board—using the same supply voltage of 1.0 V on both boards—because the current NS1e board is not instrumented to measure power and the NS1t board is not designed for high throughput. Total TrueNorth power is the sum of (*i*) leakage power, computed by measuring idle power on NS1t and scaling by the fraction of the chip's cores used by the network, and (*ii*) active power, computed by measuring total power during classification on NS1t, subtracting idle power, and scaling by the classification throughput (FPS) measured on NS1e.[#] Our focus was to characterize operation on the TrueNorth chip as a component in a future embedded system. Such a system will also need to consider capabilities and energy requirements of sensors, transduction, and off-chip communication, which requires hardware choices that are application specific and are not considered here.

**Performance.** Table 3 and Fig. 4 show our results for all eight datasets and a comparison with state-of-the-art approaches, with measured power and classifications per energy (FPS per Watt) reported for single-chip networks. It is known that augmenting training data through manipulations such as mirroring can improve scores on test data, but this adds complexity to the overall training process. To maintain focus on the algorithm presented here, we do not augment our training set and therefore compare our results to other works that also do not use data augmentation. Our experiments show that for almost all of the benchmarks, a single-chip network is sufficient to come within a few percent of state-of-the-art accuracy. Increasing to up to eight chips improved accuracy by several percentage points, and in the case of the voice activity detection (VAD) dataset, surpassed state-of-the-art performance.

## Discussion

Our work demonstrates that the structural and operational differences between neuromorphic computing and deep learning are not fundamental and points to the richness of neural network constructs and the adaptability of backpropagation. This effort marks an important step toward a new generation of applications based on embedded neural networks.

These results help to validate the neuromorphic approach, which is to provide an efficient yet flexible substrate for spiking neural networks, instead of targeting a single application or network structure. Indeed, the specification for TrueNorth and a prototype chip (39) were developed in 2011, before the recent resurgence of convolutional networks in 2012 (25). Not only is TrueNorth capable of implementing these convolutional networks, which it was not originally designed for, but it also supports a variety of connectivity patterns (feedback and lateral, as well as feedforward) and can simultaneously implement a wide range of other algorithms (6, 13, 15, 40–42). We envision running multiple networks on the same TrueNorth chip, enabling composition of end-to-end systems encompassing saliency, classification, and working memory. In this way, TrueNorth is notably different from recently proposed hardware architectures such as (43, 44), which are specifically designed to implement convolution operations.

[¶]Additional network sizes for the audio datasets (VAD, TIMIT classification, TIMIT frames) were created by adjusting features per layer or removing layers.

[#]Active energy per classification does not change as the chip's tick runs faster or slower as long as the voltage is the same (as in the experiments here) because the same number of transistors switch independent of the tick duration.

We see several avenues of potentially fruitful exploration for future work. Several recent innovations in unconstrained deep learning that may be of value for the neuromorphic domain include deeply supervised networks (34) and modified gradient optimization rules. The approach used here applies hardware constraints from the beginning of training, that is, constrain-then-train, but innovation may also come from constrain-while-train approaches, where training initially begins in an unconstrained space, but constraints are gradually introduced during training (12). Finally, codesign between algorithms and future neuromorphic architectures promises even better accuracy and efficiency.

**ACKNOWLEDGMENTS.** We acknowledge Rodrigo Alvarez-Icaza for support with hardware infrastructure. This research was sponsored by the Defense Advanced Research Projects Agency under Contract FA9453-15-C-0055. The views, opinions, and/or findings contained in this paper are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the US Government.

1. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *Nature* 323:533–536.
2. Fukushima K (1980) Neocognitron: A self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern* 36(4): 193–202.
3. LeCun Y, et al. (1989) Backpropagation applied to handwritten zip code recognition. *Neural Comput* 1(4):541–551.
4. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444.
5. Mead C (1990) Neuromorphic electronic systems. *Proc IEEE* 78(10):1629–1636.
6. Merolla PA, et al. (2014) A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345(6197):668–673.
7. Lin M, Chen Q, Yan S (2014) Network in network. *International Conference on Learning Representations (ICLR)*. arXiv:1312.4400v3.
8. Bartol TM, et al. (2015) Nanoconnectomic upper bound on the variability of synaptic plasticity. *eLife* 4:e10778.
9. Jin J, Dundar A, Culurciello E (2014) Flattened convolutional neural networks for feedforward acceleration. arXiv:1412.5474.
10. Stromatias E, et al. (2015) Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms. *Front Neurosci* 9:222.
11. Courbariaux M, Bengio Y, David JP (2015) Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in Neural Information Processing Systems* 28:3105–3113.
12. Wu Z, Lin D, Tang X (2015) Adjustable bounded rectifiers: Towards deep binary representations. arXiv:1511.06201.
13. Diehl PU, et al. (2016) Truehappiness: Neuromorphic emotion recognition on truenorth. arXiv:1601.04183.
14. Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems* 28: 1135–1143.
15. Esser SK, Appuswamy R, Merolla P, Arthur JV, Modha DS (2015) Backpropagation for energy-efficient neuromorphic computing. *Advances in Neural Information Processing Systems* 28:1117–1125.
16. Courbariaux M, Bengio Y (2016) Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv:1602.02830.
17. Bell AJ, Sejnowski TJ (1997) The "independent components" of natural scenes are edge filters. *Vision Res* 37(23):3327–3338.
18. Vedaldi A, Lenc K (2015) MatConvNet: Convolutional neural networks for MATLAB. *Proceeding of the 23rd ACM International Conference on Multimedia* (ACM, New York).
19. Amir A, et al. (2013) Cognitive computing programming paradigm: A corelet language for composing networks of neurosynaptic cores. *Neural Networks (IJCNN), The 2013 International Joint Conference* (IEEE, Dallas), pp 1–10.
20. Painkras E, et al. (2013) Spinnaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J Solid-State Circuits* 48(8):1943–1953.
21. Pfeil T, et al. (2013) Six networks on a universal neuromorphic computing substrate. *Front Neurosci* 7:11.
22. Moradi S, Indiveri G (2014) An event-based neural network architecture with an asynchronous programmable synaptic memory. *IEEE Trans Biomed Circuits Syst* 8(1): 98–107.
23. Park J, Ha S, Yu T, Neftci E, Cauwenberghs G (2014) A 65k-neuron 73-Mevents/s 22-pJ/event asynchronous micro-pipelined integrate-and-fire array transceiver. *Proceedings of the Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE* (IEEE, Lausanne, Switzerland), pp 675–678.
24. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.
25. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* 25: 1097–1105.
26. Krizhevsky A (2009) *Learning Multiple Layers of Features From Tiny Images* (University of Toronto, Toronto).
27. Netzer Y, et al. (2011) Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011* (NIPS, Granada, Spain).
28. Stallkamp J, Schlipsing M, Salmen J, Igel C (2012) Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Netw* 32:323–332.
29. Romberg S, Pueyo LG, Lienhart R, Zwol RV (2011) Scalable logo recognition in real-world images. *Proceedings of the 1st ACM International Conference on Multimedia Retrieval* (ACM, New York), p 25.
30. Garofolo J, et al. (1993) *TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1* (Linguistic Data Consortium, Philadelphia).
31. Varga A, Steeneken HJM (1993) Assessment for automatic speech recognition II: NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems. *Speech Commun* 12(3):247–251.
32. Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M (2014) Striving for simplicity: The all convolutional net. arXiv:1412.6806.
33. Preissl R, et al. (2012) Compass: A scalable simulator for an architecture for cognitive computing. *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (IEEE Computer Society Press, Los Alamitos, CA), p 54.
34. Lee CY, Xie S, Gallagher P, Zhang Z, Tu Z (2014) Deeply-supervised nets. arXiv: 1409.5185.
35. Cireşan D, Meier U, Masci J, Schmidhuber J (2012) Multi-column deep neural network for traffic sign classification. *Neural Netw* 32:333–338.
36. Pham TV, Tang CT, Stadtschnitzer M (2009) Using artificial neural network for robust voice activity detection under adverse conditions. *Proceedings of the International Conference on Computing and Communication Technologies* 1:1–8.
37. Chang HA, Glass JR (2007) Hierarchical large-margin gaussian mixture models for phonetic classification. *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding* (IEEE, Kyoto).
38. Graves A, Jaitly N, Mohamed A (2013) Hybrid speech recognition with deep bidirectional lstm. *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding* (IEEE, Olomouc, Czech Republic), pp 273–278.
39. Merolla P, et al. (2011) A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. *Proceedings of the IEEE Custom Integrated Circuits Conference* (IEEE, San Jose, CA), pp 1–4.
40. Esser SK, et al. (2013) Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores. *Proceedings of the Neural Networks (IJCNN), The 2013 International Joint Conference* (IEEE, Dallas), pp 1–10.
41. Diehl PU, Zarrella G, Cassidy A, Pedroni BU, Neftci E (2016) Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. arXiv:1601.04187.
42. Das S, et al. (2015) Gibbs sampling with low-power spiking digital neurons. *Proceedings of the IEEE International Symposium on Circuits and Systems* (IEEE, Lisbon, Portugal).
43. Conti F, Benini L (2015) A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters. *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition* (IEEE, Grenoble, France), pp 683–688.
44. Chen YH, Krishna T, Emer J, Sze V (2016) 14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *Proceedings of the 2016 IEEE International Solid-State Circuits Conference (ISSCC)* (IEEE, San Francisco), pp 262–263.