Understanding the Design of IBM Neurosynaptic System and Its Tradeoffs: A User Perspective

Hsin-Pai Cheng, Wei Wen, Chunpeng Wu, Sicheng Li, Hai (Helen) Li and Yiran Chen
Electrical and Computer Engineering Department, University of Pittsburgh, Pittsburgh, PA 15261
{hsc38, wew57, chw127, sil27, hal66, yic52}@pitt.edu

ABSTRACT

As a large-scale commercial spiking-based neuromorphic computing platform, IBM TrueNorth processor received tremendous attentions in society. However, one of the known issues in TrueNorth design is the limited precision of synaptic weights. The current workaround is running multiple neural network copies in which the average value of each synaptic weight is close to that in the original network. We theoretically analyze the impacts of low data precision in the TrueNorth chip on inference accuracy, core occupation, and performance, and present a probability-biased learning method to enhance the inference accuracy through reducing the random variance of each computation copy. Our experimental results proved that the proposed techniques considerably improve the computation accuracy of TrueNorth platform and reduce the incurred hardware and performance overheads. Among all the tested methods, L1TEA regularization achieved the best result, say, up to 2.74% accuracy enhancement when deploying MNIST application onto TrueNorth platform. In May 2016, IBM TrueNorth team implemented convolutional neural networks (CNN) on TrueNorth processor and coincidently use a similar method, say, trinary weights, {-1, 0, 1}. It achieves near state-of-the-art accuracy on 8 standard datasets. In addition, to further evaluate TrueNorth performance on CNN, we test similar deep convolutional networks on TrueNorth, GPU and FPGA. Among all, GPU has the highest throughput. But if we consider energy consumption, TrueNorth processor is the most energy-efficient one, say, > 6000 frames/sec/Watt.

1. INTRODUCTION

Human brain can process multiple tasks simultaneously with very low energy consumption. It is composed of 100 billion nerve cells with the ability to gather, transit and process signals. Inspired by the fact that human brain is much more efficient than any nowadays computers, neuromorphic computing aims performing near human brain's ability of processing a huge amount of data in an extremely short time. The design of the corresponding software and hardware is also developed for highly parallel non-von Neumann architectures. Neuromorphic computing can be performed on various platforms such as ASICs, FPGA, etc. For example, VLSI systems that mimic neuro-biological architecture is an alternative solution of von Neumann architecture in future computing systems. The concept of neuromorphic computing is also extended to the systems facilitating the computation of neural network and machine learning algorithms [1]. Very recently, IBM Neurosynaptic system was released as a well-known project that dedicated to energy-efficient neural network applications [2].

IBM Neurosynaptic system includes a TrueNorth chip, a programming environment, a 1-1 simulator, and other development APIs. From the hardware side, TrueNorth chip is a special ASIC neuromorphic computing chip that contains 5.4 billion transistors with 4,096 neurosynaptic cores, 1 million digital neurons, and 256 million synapses. It is structured with 100 billion parallel computational units (neurons) [3][14]. TrueNorth chip can achieve a peak computation of 58 Giga synaptic operations per second with only a power consumption of mere 65mW, making it extremely efficient and promising in neuromorphic computing applications.

From the software side, Convolutional Neural Network (CNN) is widely adopted in many deep learning applications. CNN gained significant success in solving visual and audio recognition problems, e.g., handwritten digit recognition and large-scale image and audio classification. The numbers of new machine learning algorithms grow rapidly. IBM TrueNorth chip has been proved to be able to perform DNN very efficiently and achieved near state-of-the-art performance on eight standard datasets including image and speech classifications. Take CIFAR100 dataset as an example, 65.48% accuracy can be reached with 31492 neurosynaptic cores, or about 8 TrueNorth chips.

As the neural networks go deeper, the accuracy of deep learning algorithms reaches a very high level: For instance, Kaiming He *et al.* [4] used residual networks with a depth of 152 layers and obtained 3.57% error rate on the ImageNet test. To achieve such a high accuracy on high resolution datasets, it is necessary to use 8 bit of neuron states. Hence, even though TrueNorth achieves a high efficiency on machine learning acceleration, there exist some remaining issues such as high core occupancy and the gap between state-of-the art accuracy. For the accuracy part, all communications between the neurosynaptic cores are represented by binary spikes. And the synaptic weights are stored as integers and be selected by lookup table. The low resolution of the data representation induces quantization loss when mapping a trained neural network in floating-point data format onto a TrueNorth chip and therefore, results in inference accuracy degradation. To compensate the quantization loss, stochastic computing methods are adopted: data is statistically represented in both temporal and spatial domains, i.e., using multiple spikes or hardware copies of the neural networks [5]. The out-

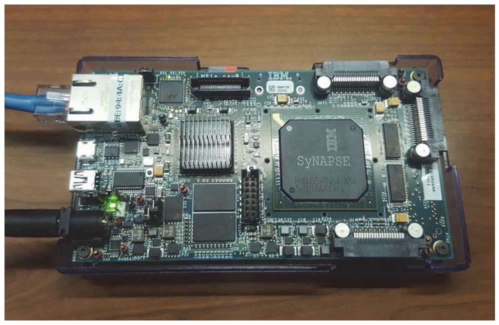


Figure 1. NS1e development platform with a single IBM TrueNorth processor.

come is obtained by averaging the results of the redundant computations. This workaround prolongs the execution time and increases the required hardware resources [6]. Our team and IBM TrueNorth team have tried to solve this issue coincidently. In this paper, we will show that we alleviate the issue by biasing the weight to digital value using regularization function in training phase. Coincidently, IBM TrueNorth team also adopted trinary weights in their implementation of CNN. The authors luckily be one of the first a few users that received the TrueNorth chip in the world and could perform research and study on it. As a summary of our contributions, we first systematically analyze the progress of IBM Neurosynaptic System. We then analyze quantization loss and introduce our regularization method. Finally, we compare the CNN performance on TrueNorth and conventional platforms.

2. PRELIMINARY

Until now, IBM has released three types of TrueNorth hardware systems, NS1e, NS1e-16 and NS16e. NS1e is the first edition of TrueNorth platform. It contains a single TrueNorth chip which provides 1 million neurons and 256 million synapses as well as sensors. NS1e-16 literally means it contains 16 NS1e boards. These 16 NS1e boards are connected with a gateway server and integrated into a rack, while the processors are not interconnected with each other. However, the newest platform, NS16e, is different. This platform is using the characteristic of chip-to-chip asynchronous communication interfaces. These 16 processors are interconnected; the spikes can perform inter-chip communications.

Figure 2 depicts the architecture of TrueNorth chip. The chip is composed of a scalable network of configurable neurosynaptic cores [7], each of which contains memory ("synapses"), processors ("neurons"), and communication ("axons"). The inter-core communication is carried by all-or-none spike events over a message-passing network. The numbers of the neurons and the axons in a neurosynaptic core are limited at 256 and one axon in a core can connect to only one core. The TrueNorth chip is physically designed to efficiently implement regular neural networks, which must be mapped to densely connected 256×256 local proximities.

A TrueNorth chip is made of a network of neurosynaptic cores. Each core includes a 256×256 configurable synaptic crossbar connecting 256 axons and 256 neurons in close proximity. The inter-core communication is transmitted by spike events through axons. Determined by the axon type at each neuron, the weight of the corresponding synapse in the crossbar is selected from 4 possible integers.

As illustrated in Figure 2 (a), in an artificial neural network, the output of a neuron (z) is determined by the input vector ($\mathbf{x}$) and the weights of the connections to the neuron ($\mathbf{w}$), such as

$$\mathrm{y} = \mathbf{w} \cdot \mathbf{x} + \mathrm{b}, \text{ and} \tag{1}$$

$$\mathrm{z} = h(y). \tag{2}$$

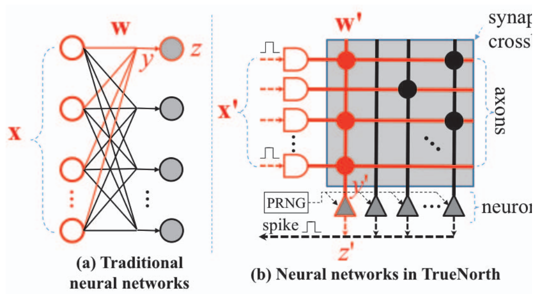

w
y
z
x
w'
synap
cross
x'
axons
y'
PRNG
spike
neuro
z'
(a) Traditional neural networks
(b) Neural networks in TrueNorth

Figure 2. Mapping neural networks in IBM TrueNorth.

Here b is a bias. $h(\cdot)$ is a nonlinear activation function. The data in (1) and (2) usually are represented with floating-point precision.

When mapping the neural network to a neurosynaptic core in TrueNorth, as shown in Figure 2 (b), the input vector $\mathbf{x}$ is quantized into the spike format $\mathbf{x}'$ and the weights $\mathbf{w}$ are approximated by random synaptic weight samples $\mathbf{w}'$. More specific, a pseudo random number generator (PRNG) is used to sample each connection in the synaptic crossbar: when the synapse is sampled to be "ON", the value indexed by the axon type will be applied as its weight; otherwise the synapse is disconnected. For the neuron implementation, rather than using nonlinear activation functions in traditional neural networks, TrueNorth adopts a spiking neuron model − *leaky integrate-and-fire* (LIF) model [8] to produce output in digital spike format. The model has 22 parameters (14 of which are user configurable) and 8 complex neuron specification equations. Regardless the possible versatility and complexity of LIF designs, a simple version of LIF, *e.g.*, McCulloch-Pitts neuron model, can be used to realize many high-level cognitive applications. The operations of McCulloch-Pitts can be mathematically formulated as:

$$y' = \mathbf{w}' \cdot \mathbf{x}' - \lambda, \text{ and} \tag{3}$$

$$z' = \begin{cases} 1, reset\ y' = 0; if\ y' \geq 0 \\ 0, reset\ y' = 0; if\ y' < 0 \end{cases}. \tag{4}$$

Here, λ is a constant leak. Note that the spiking time is omitted in (3) and (4) because McCulloch-Pitts model is history-free.

The implementation of TrueNorth differs from traditional artificial neuron model in the following three aspects: First, the inputs ($\mathbf{x}'$) and the neuron output (z') are represented in a binary format of spikes; second, the weights ($\mathbf{w}'$) are the integers selected from 4 synaptic weights, and the selections are determined by the axon type at the corresponding neurons [8]; finally, a digital and reconfigurable LIF model is used to represent the neuron function.

Such a digitized quantization operation inevitably results in system accuracy degradation. TrueNorth attempts to compensate this loss by introducing stochastic computation processes in temporal and spatial domains: first, several neural codes are supported and used to provide a series of input spikes that follow the probabilities of the normalized input value (*e.g.*, pixel) intensities; second, the PRNG in LIF controls the synapse connectivity and makes the average value of the synaptic weight close to the one in the original neural network model with a high precision. As such, sufficient accuracy can be promised by averaging the computation results over all the spike samples in a long temporal window as well as a large number of the network instantiations of the neurosynaptic cores.

3. METHODOLOGY

However, temporal spike sampling method in TrueNorth significantly slows down the cognitive inference. In addition, the duplicated network copies will rapidly occupy the available cores as the scale of neural network model raises. In observation of these drawbacks of the TrueNorth solution, in this work, we propose *a synaptic connectivity probability-biased learning method* to speedup cognitive inference, reduce core occupation, and improve inference accuracy of the neural network deployed on TrueNorth.

Based on the above observation and analysis, we tried seven different regularization methods as shown in Figure 3. And we found that the triangular shape regularization function, we named it L1TEA, has the best performance. Therefore, we propose L1TEA as a new learning method which biases synaptic connectivity probability to deterministic boundaries (*i.e.*, $p_i = 1$ or $p_i = 0$) and therefore improves the inference accuracy of TrueNorth implementa-

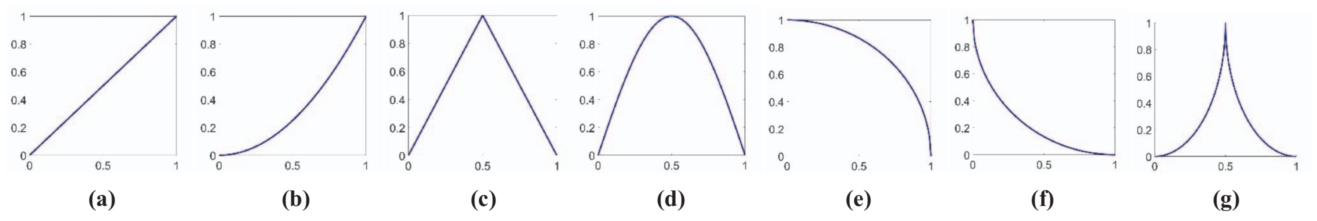


Figure 3. The seven regularizations (penalty curves). (a) L1-norm (b) L2-norm (c) L1TEA (d) Sine function (e) Circle_00 (f) Circle_11 (g) Circle_0111. The *x*-axis is the probability p_i. The *y*-axis is the penalty value.

tion. When training a TrueNorth model with floating-point precision, biasing the connectivity probability **p** is equivalent to tuning the weights **w**. Thus, the connectivity probability and the weights can be interchangeable in this context.

In the back-propagation training of neural networks [13], an effective technique to enforce constraints on weights is to add weight penalty to the minimization target function, such as

$$\hat{E}(\mathbf{w}) = E_D(\mathbf{w}) + \lambda \cdot E_W(\mathbf{w}) \tag{5}$$

Here **w** is the vector of all weights in the neural network; $E_D(\mathbf{w})$ is the inference error to be minimized; $E_W(\mathbf{w})$ is the weight penalty; and λ is a regularization coefficient. An efficient penalty to bias **w** to zeros is *L1-norm*, which is $E_W(\mathbf{w}) = \| \mathbf{w} \| = \sum_{k=1}^{M} | w_k |$.

Our experiments on MNIST show that in all three layers of a feed-forward neural network, 88.47%, 83.23% and 29.6% of weights can be zeroed out, with slight classification accuracy drop from 97.65% to 96.87%. The neural network has two hidden layers composed of 300 and 100 hidden neurons, respectively.

Although *L1-norm* is an effective method to bias weights to zeros, it doesn't help to reduce synaptic variance. For example, for the test bench of MNIST handwritten digit recognition, the histogram distributions of the connectivity probability (weight) without any penalty function and the one after applying *L1-norm* penalty are presented in Figure 4 (a) and (b), respectively. The recognition accuracies of the two designs are very close (i.e., 95.27% vs. 95.36%). Interestingly, the experiment shows that a large portion of the weights fall near the poles (p_i = 1 or 0) in Figure 4 (a) even without any penalty. *L1-norm* can further bias the probability distribution to zero but pushes it away from the deterministic pole of p_i = 1. Thus, after deploying the weights in Figure 4 (b) on the four neurosynaptic cores of TrueNorth, the accuracy degrades to 89.83%.

The above observation inspired us to use a probability-biased learning method that can aggressively bias the probability histogram toward the two poles (and away from the worst point at centroid) to improve the inference accuracy. In particular, the proposed scheme uses the following penalty function to bias the histogram toward the two poles:

$$E_b(\mathbf{w}) = \| \, |\mathbf{w} - a| - b \, \| = \sum_{k=1}^{M} \big| \, |w_k - a| - b \, \big|, \tag{6}$$

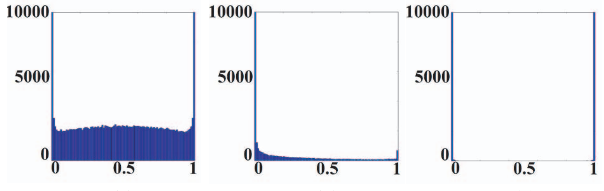


Figure 4. Probability (weight) distribution under L1Norm and L1TEA.

where a denotes the centroid, it biases away and b is the distance between the poles and the centroid. Note that if we represent entire |**w**-a|-b with variable **s**, the biasing penalty $E_b(\mathbf{w})$=||**s**|| becomes a *L1-norm* penalty, which tends to pull **s** toward zeros. Hence, the purpose of the biasing penalty function in Equation (6) is to pull **w** to $a + b$ and $a - b$. A special case is $a = b = 0.5$ that pushes connectivity probabilities to the poles at the two ends by enforcing the highest penalty on the worst case and the lowest (none) penalty on the best cases.

Figure 4 (c) shows the learned probability histogram after applying our proposed biasing penalty function by setting $a = b = 0.5$. The obtained recognition accuracy is 95.03%, which is slightly lower than the 95.27% of the network without any penalty. Nevertheless, this new model reaches an accuracy of 92.78% after being deployed to TrueNorth with connectivity quantization. Comparing to the previously received accuracy of 90.04% (89.83%) without any penalty (with *L1-norm* penalty), our learning method raises the recognition accuracy by 2.5% (2.71%) using the same number of the synaptic cores. The improvement comes from the fact that almost all connective probabilities of synapses are biased to the deterministic states (0 or 1), as illustrated in Figure 4 (c). It implies a minimized synaptic variance. More evaluations on the hardware resource reduction, performance, and accuracy that achieved by our learning method will be presented Section in detail.

To illustrate the efficacy of our method, *e.g.*, the synaptic variance in neurosynaptic cores, the maps of synaptic weight deviation between the desired learned TrueNorth model and the deployed TrueNorth model are extracted from IBM Neuro Synaptic Chip Simulator (NSCS) and plotted in Figure 5. In the figures, each deviation map corresponds to a randomly selected neurosynaptic core in a TrueNorth chip, and each point in the map (256x256 points in total) is the deviation of deployed synaptic weight from the desired weight with floating-point precision (note that the deviation is normalized by the maximum possible synaptic weight). Figure 5 (a) and (b) are the deviations obtained with biasing optimization and without any penalty optimization, respectively. Without applying our learning method, the deviation in Figure 5(b) is large, say, 24.01% of the synapses having deviations larger than 50%. After applying our probability-biased learning, 98.45% of the synapses

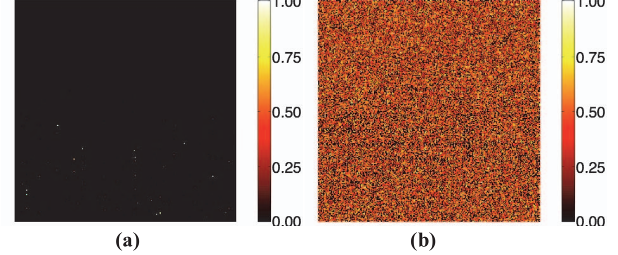


Figure 5. (a) is the synaptic weight deviation of deployment by probability-biased learning, comparing with that of Tea learning in (b).

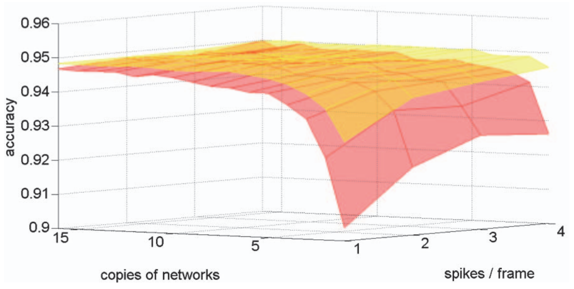


Figure 6. Absolute accuracies of Tea learning (red lower surface) and probability-biased learning (yellow upper surface).

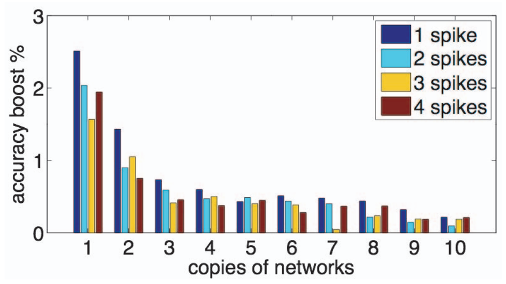


Figure 7. Accuracies boost (our method minus Tea learning).

have zero deviation, as shown in Figure 5 (a). In fact, less than 0.02% of the synapses have deviations larger than 50%.

4. EXPERIMENT RESULT

We use datasets in Table 1 for our classification experiments, including MNIST handwritten digits [9] and RS130 protein secondary structure dataset [10]. Protein secondary structure is the local conformation of the polypeptide chain and classified to three classes: alpha-helices, beta-sheets, and coil.

4.1 Accuracy Efficiency

First, we show the efficiency of our probability-biased learning method to retain the accuracy. Figure 6 illustrates resultant accuracies from various combinations of network (1 to 16 copies) and spike (1 to 4 spikes per frame) duplication. Here spike per frame (*spf*) is defined as the number of spike samples utilized to encode each pixel/activation. The red and yellow surface is the accuracy surface of Tea learning and our probability-biased learning, respectively. From Figure 6, we can see both spatial and temporal duplication can reduce accuracy loss caused by low quantitation precision. However, as duplications increase, the accuracy surfaces saturate to the plane limited by the original accuracy trained by Caffe (95%) [11]. Note that, we have averaged accuracy at each grid over ten results considering the randomness in TrueNorth.

We can see that our accuracy (yellow) surface covers above the original (red) one from Tea learning. It shows that under the same spatial and temporal duplications, our method retains better accuracy (especially when the number of duplication is small).

Figure 7 depicts the accuracy improvements achieved by our learning method w.r.t. Tea learning. The highest gain (2.5%) is reached at the lowest levels of duplication (one network copy and one *spf*). In general, when fewer copies of networks are occupied (which means more cores are saved), a higher accuracy improvement can be achieved, showing less reliance of our approach on spatial duplication.

4.2 Core Occupation Efficiency

To evaluate efficiency of our method in reducing resource consumption, we compare core occupation required for the same accuracy when utilizing our method versus Tea learning. Note that it is difficult to achieve exactly equivalent accuracy between two TrueNorth neural networks. Nonetheless, we show that even when adopting a comparison procedure that is biased toward Tea learning, our method still requires less cores. Be specific, the biased nature of our comparison occurs when the same accuracy cannot be obtained between the two compared methods – in such a case, we select the accuracy of the Tea learning as the baseline and compare it to the network required for the next greater level of accuracy when using our method. Table 2 (a) shows the comparison of the occupied cores between two tested TrueNorth neural networks and their corresponding accuracies. Accuracies are ordered in ascend and two types of neural networks with nearest accuracies are grouped (with gray shadow) in the table for easy comparison. In each group, the accuracy achieved by our method is either equal or higher than that of the current method. The number (and the percentage) of saved cores is recorded in the third row. These networks are denoted by N# and B#, where N/B indicates the model is learned without penalty (**N**one) or with **B**iasing penalty and # is the number of copies of the network.

Under 1 *spf* shown in Table 2 (a), our approach can save on average 49.5% cores with equivalent (or higher) accuracy. Moreover, the number of saved cores increases with the desired level of accu-

Table 1. Test datasets

Dataset	Description	Area	Training size	Testing size	Feature #	Class #
MNIST	Handwritten digits	Computer Engineering	60,000	10,000	784 (28×28)	10
RS130	Protein secondary structure	Life Science	17,766	6,621	357	3

Table 2. Core occupation and performance efficiency of probability-biased learning method

(a) Core occupation efficiency (1 *spf*)

Accuracy	**0.904**	**0.924**	**0.929**	**0.935**	**0.938**	**0.939**	**0.942**	**0.942**	**0.943**	**0.944**	**0.945**	**0.946**	**0.947**	**0.947**
Network Copies	N1*	N2	B1	N3	B2	N4	N5	B3	N7	N9	B4	N10	N16	B5
Saved Core	-	4 (50.0%)		4 (33.3%)		-	8 (40.0%)		-	20 (55.6%)			44 (68.8%)	

(b) Performance efficiency (1 network copy)

Accuracy	**0.904**	**0.920**	**0.927**	**0.928**	**0.929**	**0.932**	**0.933**	**0.934**	**0.940**	**0.943**	**0.946**	**0.947**	**0.948**	**0.950**
spf	N1*	N2	N3	N6	B1	N7	N11	N13	B2	B3	B4	B5	B9	B13
Speedup				6				6.5						

* TrueNorth neural networks are denoted by N# and B#, where N/B indicates the model is learned without penalty (**N**one)/**B**iasing penalty. # is the number

racy, showing the scalability of our method to higher inference accuracy, which is often the highest-priority demand for intelligent systems.

4.3 Performance Efficiency

We also evaluate the performance/speed efficiency of our learning method. The results are listed in Table 2 (b), which has a similar format to that of Table 2 (a) except that here # denotes the number of *spf*. With one copy of the neural network, our method, for example B2, can still achieve 0.60% higher accuracy with a speed of 2 *spf*, comparing to 13 *spf* (N13) using Tea learning method. This result can be translated to 6.5× speedup and similar results are also observed at different instantiated network copies.

5. FROM TEA LEARNING TO EEDN

The above experiments also show that we can implement a neural network by using the Tea learning layer on TrueNorth that has a function similar to multilayer perceptron. In all the experiments that we performed using the Tea layer, only 16 to 64 cores are occupied.

However, the real-world applications can be much more complex. Take CIFAR10 as an example, CIFAR10 is consist of 32x32 color images in 10 classes. We have 3 channels, RGB, and much more pixels for each image. Hence, we need 3072 (32x32x3) input neurons for each image. It becomes hard for using Tea learning to perform classification. Therefore, TrueNorth team proposed *Energy-efficient deep neuromorphic networks* (Eedn) [15], which can efficiently implement CNN for TrueNorth. In Eedn, the weight distribution problem is also improved. They use two synapse to represent trinary weights ({-1, 0, 1}). Besides, the sparsity is also combined in the training phase by adding $\gamma\frac{1}{2}\sum\bar{y}^2$ to the cost function.

5.1 User perspective

From 2015 to 2016, the end-to-end workflow basically remains the same. It includes data collection, feature extraction, network training, model building, and deployment on TrueNorth. The evolutional innovation is the change from Corelet Programming Environment 2.1 (cpe2.1) to Corelet Programming Environment 2.2 (cpe2.2). cpe 2.2 has more features such as CNN implementation, trinary weights, sparsity network structure, etc. And it is more convenient that the training and building phases are combined in the MATLAB. On the other hand, it lacks flexibility. Each types of layers of TrueNorth is well written but encrypted to a protected function file. As a user, there's no way to access and modify the details in the convolutional layer such as regularization, activation function, etc.

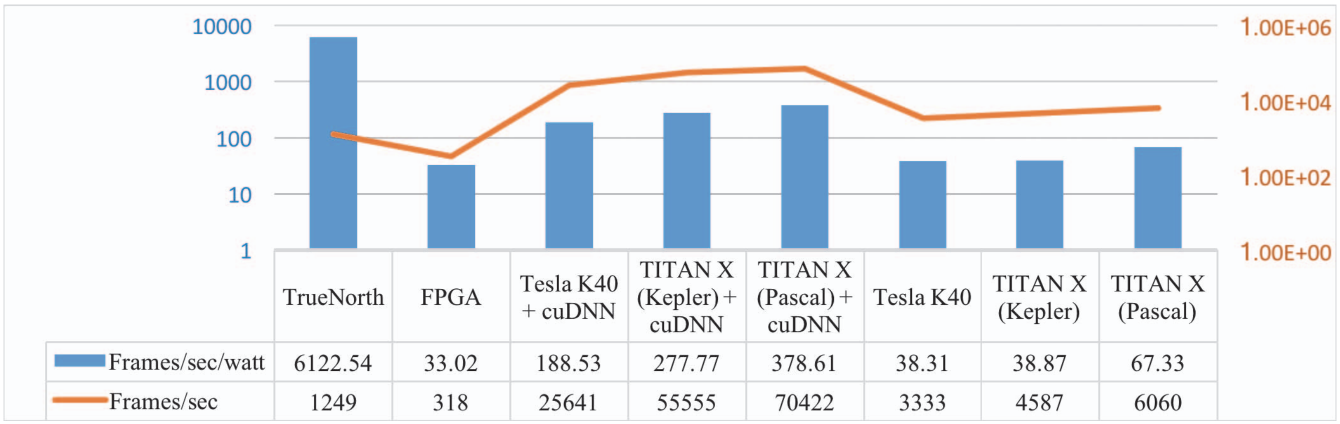
Table 3. The comparison of cpe2.1 and cpe2.2

	cpe2.1	*cpe2.2*
Training framework	Caffe	MatConvNet
Flexibility	More	less
Weight distribution	Approximate by 4 digital value	Biased to {-1, 0,1}
Sparsity	Tea learning	regularization in cost function

Table 4. The comparison of core occupancy and accuracy on TrueNorth Eedn

	Accuracy (%)	**Layer #**	**Feature map #**	**Core occupancy**
MNIST	99.5	15	2048	3098
	99.08	6	512	1342
CIFAR10	83.41	15	2048	4064
	82.58	6	512	2560

It is indeed a breakthrough that TrueNorth can perform CNNs. It also indicates that TrueNorth is a flexible platform to construct different types of network structure. On the contrary, achieving a high accuracy in TrueNorth is still core consuming. As shown in Table 4, the best classifications on MNIST and CIFAR10 that a single TrueNorth chip can achieve are 99.5% and 83.41%, respectively, where 15 layers were used including preprocessing layers, dropout layers, and convolutional layers. Increasing the numbers of feature maps and layers will correspondingly increase the core occupancy. Achieving a marginal accuracy enhancement may require significantly more cores. For example, if we eliminate 9 convolutional layers and only use the necessary layers, the accuracy only drop 0.42% and 0.83% on MNIST and CIFAR10, respectively. But the core occupancy is significantly improved, say, saving up to 1756 cores in MNIST. Our understanding is that, TrueNorth can easily perform approximated result at this moment. But if we want to mitigate the gap between approximated accuracy and state-of-the-art one, the number of cores that should be used will sharply increase, which is the intrinsic tradeoff when designing the network structure in TrueNorth.



	TrueNorth	FPGA	Tesla K40 + cuDNN	TITAN X (Kepler) + cuDNN	TITAN X (Pascal) + cuDNN	Tesla K40	TITAN X (Kepler)	TITAN X (Pascal)
Frames/sec/watt	6122.54	33.02	188.53	277.77	378.61	38.31	38.87	67.33
Frames/sec	1249	318	25641	55555	70422	3333	4587	6060

Figure 8. Performance comparison in FPS and Frames/sec/watt

5.2 CNN performance Comparison among TrueNorth, GPU and FPGA

To have a more straightforward analysis on TrueNorth's performance, we compare the performance of TrueNorth, GPU, and FPGA [12]. By comparing Frames Per Second (FPS), and Frames/Second/Watt, we can have a further understand of the throughput and energy efficiency of each platform.

The dataset we use is CIFAR10. The TrueNorth board we use is NS1e. The FPGA we use is Zed board, which is powered by the Xilinx Zynq-7000 (APSoC). The FPGA-based CNN accelerator is designed with Vivado HLS, while the hardware synthesis and the bitstream generation have been performed with Vivado Design Suite. The GPUs we use are GTX TITAN X (Kepler), Tesla K40, GTX TITAN X (Pascal). We also compare the performance of using CUDA Neural Network Libray (cuDNN) for GPU or not.

Since that these three platforms have different structures, to perform a fair comparison, we tune these three networks to achieve almost the same accuracy on CIFAR10, which is around 83.28% to 84.5%. The result is shown in Figure 8. Generally, the GPU has the best performance on FPS. The lowest amount is 3333 (FPS), which is much higher than any other platform. Besides, GPU with cuDNN enabled can achieve a very high FPS. For example, 55556 (FPS) is achieved by TITAN X (Kepler) with cuDNN enabled. However, GPU is also very power consuming. If we consider power consumption and performance together, TrueNorth has the highest number, say, 6122 (frame/sec/watt), Neither GPU nor FPGA is not comparable to such efficiency.

6. CONCLUSION

In this work, we theoretically analyze the impacts of low data precision in TrueNorth on inference accuracy, core occupation, and performance, and propose a probability-biased learning method to enhance the inference accuracy through reducing the random variance of each computation copy. We also evaluate energy efficiency of deploying CNN on TrueNorth by comparing with multiple types of GPUs and FPGA. We conclude that it is very core consuming for TrueNorth to achieve high accuracy but it is also very energy efficient to achieve an approximated result.

7. ACKNOWLEDGMENTS

This work is supported in part by AFRL FA8750-15-1-0176. We also would like to thank IBM for providing the IBM Neurosynaptic System.

8. REFERENCES

[1] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, N. Gi-Joon, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, pp. 1537-1557, 2015.

[2] J. Sawada, F. Akopyan, A S. Cassidy, B. Taba, M. V. Debole, P. Datta, R. Alvarez-Icaza, A. Amir, J. V. Arthur, A. Andreopoulos, R. Appuswamy, "TrueNorth Ecosystem for Brain-Inspired Computing: Scalable Systems, Software, and Applications," The International Conference for High Performance Computing, Networking, Storage and Analysis, 2016.

[3] S. K. Esser, A. Andreopoulos, R. Appuswamy, et al., "Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores," in the 2013 International Joint Conference on Neural Networks, pp. 1-10, 2013.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. arxiv, 2015.

[5] A. S. Cassidy, R. Alvarez-Icaza, F. Akopyan, et al., "Real-time scalable cortical computing at 46 giga-synaptic OPS/watt with ~100× speedup in Time-to-Solution and ~ 100,000× reduction in Energy-to-Solution," in the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 27-38, 2014.

[6] H. Cheng, W. Wen, C. Song, B. Liu, H. Li, Y. Chen, Exploring the Optimal Learning Technique for IBM TrueNorth Platform to Overcome Quantization Loss, Proceedings of IEEE/ACM International Symposium on Nanoscale Architectures, 2016.

[7] W. Wen, C.-R. Wu, X. Hu, B. Liu, T.-Y. Ho and Y. Chen, "An EDA framework for large scale hybrid neuromorphic computing systems," in the 52nd ACM/EDAC/IEEE Design Automation Conference, pp. 12, 2015.

[8] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, A. Amir, D. B. D. Rubin, F. Akopyan, E. McQuinn, W. P. Risk, and D. S. Modha, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in the 2013 International Joint Conference on Neural Networks, pp. 1-10, 2013.

[9] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," 1998.

[10] J.-Y. Wang, "Application of support vector machines in bioinformatics," National Taiwan University, 2002.

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

[12] M. Naylor, P. J. Fox, and S. W. Moore, "Managing the FPGA memory wall: Custom computing or vector processing," International Conference on Field Programmable Logic and Applications, pp. 1-6, 2013.

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

[14] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura and B. Brezzo, "A million spiking-neuron integrated circuit with a scalable communication network and interface," Science, 345(6197), pp. 668-673, 2014.

[15] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, "Convolutional Networks for Fast, Energy-Efficient Neuromorphic Computing," Proceedings of the National Academy of Sciences, vol. 113 no. 41, 2016.