

Binary Image Classification Using a Neurosynaptic Processor: A Trade-Off Analysis

William Murphy
Air Force Research Lab
william.murphy.37@us.af.mil

Megan Renz
University at Buffalo
meganren@buffalo.edu

Qing Wu
Air Force Research Lab
qing.wu.2@us.af.mil

Abstract—This paper examines the performance of two power efficient hardware implementations using deep neural networks to perform a simple image classification task. We provide the first ever examination of the accuracy-energy trade-offs of deep neural networks running on both an embedded GPU, and a neuromorphic processor. IBM’s TrueNorth is a brain-inspired event-driven neuromorphic processor. It was designed to be scalable and to consume extremely low amounts of power. NVIDIA’s Tegra K1 SoC is a mobile processor also designed with low power and a small footprint in mind. While these two chips were designed with similar constraints, the resulting architectures and performance trade-offs achieved are significantly different. On our simple image classification task Convolutional Neural Networks utilizing the Tegra K1 SoC achieve up to 89 % accuracy with a normalized accuracy per active energy, $\frac{||Acc||}{E_A}$, score of up to 24.22 on our test dataset, while Tea Networks running on the TrueNorth processor achieve less accuracy at 82%, but a better accuracy-energy trade-off with a $\frac{||Acc||}{E_A}$ score of up to 158.49.

I. INTRODUCTION

There is a strong demand to push data processing to “the edge”, or to the sensors that collect the data. This capability contributes to the larger goal of sensor autonomy, where sensors are able to act on information that they are collecting. To make processing of data on the edge a reality many design decisions need to be made based on constraints of the sensing platforms, and the data processing tasks at hand. Some of the most common constraints on these systems include: task accuracy, available power, and hardware footprint. This paper examines some performance trade-offs of a simple image classification task executed on two hardware architectures designed for low-power data processing.

After imagery data is collected it is usually streamed back to a central location to be analyzed, or stored on the sensing platform and analyzed at some later time. Both of these scenarios present problems. In the first scenario, a very high bandwidth communications network is required and there are transmission delays that may cause the data to be too old to act on by the time it is processed. In the second scenario, the data is very likely out of date and much less useful by the time it has returned to be analyzed. Analyzing data on the sensing platform can reduce the dependency on high-bandwidth data links and allow for data to be acted on before its value diminishes.

The primary objective of this work is to investigate the accuracy-energy trade-offs between two promising architectures for low-power, high-throughput image classification. We perform the first ever examination of the accuracy-energy trade-offs of deep neural networks running on both an embedded GPU, and a neuromorphic processor. The experimental results and conclusions can be used in any decision making process where a computing architecture must be chosen given

a set of operational constraints. The work also describes the algorithmic design processes used to improve the energy-accuracy trade-offs, and the metrics used to evaluate performance in this trade space.

II. HARDWARE

Each of the platforms examined here are development boards for the processors featured on them. This suggests that while the processor on each board is highly optimized for low-power computation, the boards that house these processors were optimized for ease of development. Details of each platform are given in the following sections.

A. IBM NS1e Board

The IBM NS1e board provides an evaluation platform for the TrueNorth neurosynaptic chip. The TrueNorth chip [13] contains 4,096 TrueNorth cores emulating the functionality of 1 million neurons and 256 million synapses. The chip receives input data represented with a neural coding scheme [6], which uses binary spikes encoded over time and/or space to represent non-binary information. The destination on the TrueNorth chip to which each spike will travel is also included with each spike. A program configuration is written for the TrueNorth chip by specifying connections between axons and neurons, parameters defining the behavior of neurons, and parameters defining the behavior of the synaptic connections between axons and neurons. This program receives the spike encoded input and produces spike encoded output. More information regarding the algorithm development, programming paradigm, and neuron model for TrueNorth can be found in [5], [1], and [2] respectively.

B. NVIDIA Jetson TK1 Board

NVIDIA’s Jetson board [14] provides an evaluation platform for NVIDIA’s Tegra K1 SoC. The Jetson board is the platform used by the winning team at the most recent Low Power Image Recognition Challenge [12]. The Tegra K1 SoC contains the NVIDIA Kepler GK20a GPU with 192 SM3.2 CUDA cores, and the NVIDIA 2.32 GHz ARM quad-core Cortex-A15 CPU with Cortex-A15 battery-saving shadow-core. The Jetson is built with the same features and architecture as a standard desktop GPU, meaning that CUDA code that runs on a desktop NVIDIA GPU will run in the same way on Jetson.

III. DATASET

In our binary image classification problem, the two classes we aim to distinguish between are “Ship” and “Not Ship”. In order to train our algorithms to distinguish between these classes and subsequently to evaluate how well the algorithms are able to separate these classes, we created a labeled dataset

drawn partly from the CIFAR-10 dataset [8], and partly from the ImageNet dataset [3].

The resulting dataset sampled from the above sets consists of approximately 12,000 training images and 2,000 testing images, with each of the sets containing a roughly equal amount of positive and negative samples. Finally, all of the images were resized to 64x64x3. The labels from the original datasets were converted accordingly for the binary problem.

IV. APPROACH

The sections below discuss the details of the algorithm design and experiment procedures in addition to the metrics we use to characterize accuracy and energy.

A. Tegra K1 Algorithm Design

The widely recognized state of the art algorithm for image classification accuracy is the Convolutional Neural Network (CNN) [10]. A CNN running on the Jetson board currently holds the record in low power image recognition, as defined by the metrics presented in [12]. CNNs are very well-suited for running on GPUs due to the large number of matrix operations involved in the forward path. While CNNs already hold many competition records, little has been investigated about how to efficiently and optimally design them, especially when metrics other than accuracy are of concern.

Historically, CNNs have been designed to be as accurate as possible without much consideration for other metrics such as size, weight, and power. The easiest and most common way to improve accuracy in CNNs is to increase the depth of the network, the number of layers, or the width of the network, the number of feature maps in a layer. There are a number of drawbacks in using this approach which are magnified when the computing resources are constrained.

As the size of a CNN increases, the associated computational complexity also increases very quickly. While increasing the size of the network generally improves the classification accuracy, there is no guarantee that the new layers or parameter increments are used efficiently. In other words, many of the new layers or parameter increments may contribute little to the classification task. This phenomenon was observed by the authors of GoogLeNet [15], which achieved better accuracy on the ILSVRC classification challenge than the previous state of the art, AlexNet [9], using 12X fewer parameters. GoogLeNet introduced the "Inception Module", based on the Network in Network [11] framework designed to reduce the size of the model while increasing performance. Zeiler and Fergus [17] examined AlexNet by visualizing features and reverse engineering the processes in each layer, in order to demonstrate CNNs functionality and performance.

From the lessons learned in [17] regarding architecture selection, we investigate our application by varying model complexity in order to construct a baseline CNN architecture for our data and to subsequently reduce the computational complexity and energy without significantly sacrificing accuracy.

B. TrueNorth Algorithm Design

As described in Section II-A, the TrueNorth chip processes data encoded as spikes using networks of explicitly connected and configured axons and neurons. Tea [4] is a framework that provides a workflow and tools for training neural networks that are constrained prior to learning in a manner that allows

a systematic mapping of the trained network to the TrueNorth chip. The Tea learning method enables back-propagation based training despite non-differentiable spiking neurons and discrete integer valued synapse weights. A brief overview of the process for training and deploying a Tea network is given in the following paragraphs. Please refer to [4] for more details.

A Tea network's topology is similar to a locally connected multi-layer perceptron. The topology is specified by providing an $n \times n$ block size and a stride size s at each layer. The maximum block size is constrained by the number of axons on each TN core. Since each block will eventually be mapped to a single TN core, $256 \geq n * n * numChannels$. Each layer in the Tea network is composed of $B \times B$ blocks, where B at any given layer is given by:

$$B(layer) = \frac{B(layer - 1) - n}{s} + 1$$

where $B(0)$, the data layer, is equal to 64 in our dataset.

The connections between layers are fixed prior to training using progressive mixing to provide neurons in deeper layers inputs from broader regions of the input space. Interestingly, the synaptic weights that will be mapped to the TrueNorth chip are fixed prior to training and are not learned. Instead of the weight, the Tea training process learns the probability that a particular synapse will be connected or disconnected.

The output of the Tea network training is the topology that is constructed based on the block size, stride size, and layers that are specified, with learned probabilities that each synapse will be connected or disconnected. This probabilistic model, is then converted into a binary model where the connected/disconnected states of the synapses are configured. The resulting model specifies a TN program that can be run on the TN. The neural coding scheme used for inputs to the TN chip is the stochastic rate code [4]. To encode a pixel intensity in stochastic rate code, we first rescale the intensities to the range [0,1]. Then a time window is chosen to encode the input data, and the scaled intensity is used as the probability to generate a spike in each tick of the time window. In this work, we choose a time window of 8 to encode the input data.

Similar to the CNN model, we investigate the Tea implementation by varying model size in order to construct a baseline Tea Net architecture for our data and to subsequently reduce the computational complexity and energy of our baseline by varying the number of layers, block sizes, and stride sizes without significantly sacrificing accuracy.

C. Algorithm Selection

As discussed in the previous sections, for both the CNNs and Tea Networks, we begin with a baseline algorithm that establishes satisfactory accuracy for our dataset. Once the baseline is established, we prune the network by reducing the depth and/or width in our CNNs and by reducing the depth and/or number of used cores in our Tea networks. The baseline architecture along with the different reduced configurations are shown in Table I.

The notation "C128-C64-FC2500-FC500" represents a CNN with a 128-feature-map convolutional layer as layer 1, a 64-feature-map convolutional layer as layer 2, a 2500-node fully connected layer, and a 500-node fully connected layer. The first convolutional layer has a 5x5 kernel with stride 1, and the second convolutional layer has a 3x3 kernel with stride 1. Each of the convolutional layers is followed by a 2x2 max

Mod	CNN Topologies	Tea Topologies
Base	C64-C32-FC1000-FC250	B9S5-B4S2-B2S1-B2S1
1	C64-C32-FC500-FC100	B9S5-B6S3-B2S1
2	C32-C16-FC500-FC100	B9S5-B8S2-B2S1-B2S1
3	C32-C16-FC500-FC25	B9S5-B6S3-B3S1
4	C32-C16-FC200-FC100	B8S7-B6S3-B2S1
5	C32-C16-FC200-FC25	B8S7-B5S2-B2S1-B2S1
6	C16-C16-FC500-FC100	B8S7-B3S2-B2S1-B2S1
7	C16-C8-FC500-FC100	B8S7-B5S2-B3S1
8	C8-C8-FC500-FC100	B8S4-B6S3-B3S1-B2S1
9	C4-C4-FC500-FC100	B8S4-B5S2-B4S2-B2S1
10	C4-C4-FC50-FC10	B8S4-B6S3-B4S2

TABLE I: Models

pooling layer with stride 1, and a RELU non-linearity. The notation "B9S5-B4S2-B2S1-B2S1" represents a Tea network with layer 1 block size 9 and stride 5, layer 2 block size 4 and stride 2, etc.

We present the complexity of each of these models in Tables II and III. For Tea networks, complexity is given as the number of cores that each network occupies on the TrueNorth processor. As discussed in Section IV-B each Tea "block" is written to an individual core, therefore the number of cores occupied by the network is equal to the number of blocks in each Tea network. For the CNNs evaluated in this work, we give complexity as:

$$C = 183600C_1 + 14112C_1C_2 + 784C_2 + 392C_2I_1 + 2I_1I_2 + 2I_2$$

where C_1 , C_2 , I_1 , and I_2 represent the number of feature maps in the convolutional layers and the number of nodes in the fully connected layers respectively. Each coefficient is calculated using the kernel and stride sizes in the convolutional layers as well as the dimensionality of the input to each layer. The number of cores and CNN complexity are shown simply to demonstrate how our topological changes affect the amount of computation, and therefore energy consumed.

The first metric that we report for algorithm selection is the active energy consumed per image classification. We use active energy as a way of removing the overhead of each board. This is especially important in this work because the Jetson and NS1e boards are development boards which have been designed to facilitate ease of development and therefore have additional (power consuming) components that would be eliminated in a board built for deployment. Active energy per image is given in *Joules* by

$$E_A = t(P_{total} - P_{idle})$$

where t is the theoretical time taken to classify a single image on each platform. We use theoretical times in order to ensure that we are isolating the performance of each processor. The theoretical time for a forward pass of one image through a CNN on the Jetson board is collected by using the CAFFE [7] timing tool. The theoretical time for one image to move through a Tea Network on NS1e is given by the size of the time window used to encode each frame plus the number of "ticks" taken by an input spike to move through a Tea model. Total power, P_{total} , is observed while a network is performing classification and idle power, P_{idle} , is observed while each board is booted and only components necessary for testing

are connected. P_{idle} observations are 3.7W and 3.0W for the NS1e and Jetson boards respectively.

The accuracy of each of our networks is the next reported metric. Accuracy is given by

$$Acc = (1 - \frac{\sum_{i=1}^N |label(i) - prediction(i)|}{N})$$

where N is the total number of images.

Now, in order to select optimal networks, we combine the above two metrics. Our selection metric is given by

$$\frac{||Acc||}{E_A}$$

where the normalized accuracy term, $||Acc||$, is given by

$$||Acc|| = \frac{(Acc - 0.5)}{1 - 0.5}$$

In the $||Acc||$ term, 0.5 is the probability of randomly selecting the correct class. Performing this normalization, effectively only gives credit for all accuracy that is better than a random predictor. For both the NS1e and the Jetson boards we choose the algorithm configurations that yield the maximum $\frac{||Acc||}{E_A}$. This metric is similar to criteria for selecting the best model in the LPIRC [12], however we add the normalization term since the probability of randomly guessing the correct class is much greater in our problem. The normalization term could also be used for setting a minimum threshold for accuracy at any level.

D. Platform Evaluation

Finally, in order to evaluate the performance of CNNs on Jetson against the performance of Tea networks on NS1e, we report the performance of the two best configurations for each platform over our test dataset. We characterize the performance by reporting accuracy, theoretical time to classify each image, and $\frac{||Acc||}{E_A}$.

V. RESULTS

We break the results into two main sections. First, we present the results of the algorithm selection process, and then we present the results of the platform evaluation.

A. Algorithm Selection

Tables II and III show the results of the algorithm selection experiments over our validation data set.

Model	Cores	E_A	Acc	$ Acc $	$\frac{ Acc }{E_A}$
Baseline	194	4.29E-3	0.8	0.6	139.86
Model 1	157	4.42E-3	0.78	0.56	126.70
Model 2	158	4.15E-3	0.83	0.66	159.15
Model 3	157	3.86E-3	0.74	0.48	124.22
Model 4	86	3.91E-3	0.83	0.66	168.71
Model 5	95	4.15E-3	0.81	0.62	149.51
Model 6	110	4.34E-3	0.82	0.64	147.40
Model 7	91	3.84E-3	0.78	0.56	145.83
Model 8	246	3.9E-3	0.81	0.62	158.97
Model 9	266	4.06E-3	0.81	0.62	152.86
Model 10	242	4.2E-3	0.82	0.64	152.42

TABLE II: NS1e Algorithm Selection

From Table III we select the CNNs, Model 7 and Model 10. These networks give a 175 % and a 233 % improvement in $||Acc||/E_A$, coupled with only 1% and 4% decreases in

Model	Comp.	E_A	Acc	$\ Acc\ $	$\frac{\ Acc\ }{E_A}$
Baseline	5.37E7	0.1073	0.9	0.80	7.46
Model 1	4.7E7	0.075	0.91	0.82	10.93
Model 2	1.63E7	0.0437	0.9	0.80	18.31
Model 3	1.63E7	0.0483	0.9	0.80	16.56
Model 4	1.44E7	0.0396	0.89	0.78	19.70
Model 5	1.44E7	0.0399	0.9	0.80	20.05
Model 6	9.8E6	0.044	0.9	0.80	18.18
Model 7	6.42E6	0.0378	0.89	0.78	20.63
Model 8	4.05E6	0.038	0.85	0.70	18.42
Model 9	1.85E6	0.0361	0.85	0.70	19.39
Model 10	1.04E6	0.0289	0.86	0.72	24.91

TABLE III: Jetson Algorithm Selection

accuracy respectively when compared to the Baseline CNN. These are significant improvements with respect to accuracy-energy trade-off. This suggests that our Baseline CNN needs a disproportionate number of parameters to achieve the last few units of accuracy. The E_A consumed by each of our selected CNNs is less than half that of the Baseline CNN. This can be attributed to faster classifications and fewer operations both leading to less energy consumption.

From Table II we select the Tea networks, Model 2 and Model 4. These networks give 10 % and 13 % improvements in $\frac{\|Acc\|}{E_A}$ when compared to the Baseline Tea Network. Most of the variation in energy seen in Table II can be attributed to variation in time due to smaller networks. Since each of these Tea networks only consume 2 – 6% of the cores on the TN chip and the same input encodings are used for each network, the power drawn during each of the Tea network runs is mostly unchanged. This improvement can be attributed to both an increase in accuracy and a decrease in the time taken to classify all of the images in the validation set.

B. Platform Evaluation

Table IV shows the platform evaluation using the two best performing algorithms on each platform over our test set.

CNN on Jetson Evaluation				Tea on NS1e Evaluation			
Mod	Acc	$\frac{T}{I_m}$	$\frac{\ Acc\ }{E_A}$	Mod	Acc	$\frac{T}{I_m}$	$\frac{\ Acc\ }{E_A}$
Base	0.89	29	7.27	Base	0.8	13	139.86
7	0.87	18	19.58	2	0.82	13	154.33
10	0.85	17	24.22	4	0.81	12	158.49

TABLE IV: Platform Evaluation

Table IV shows the final evaluation of the top performing models on each platform. It can be seen that the best classification accuracy is achieved by a CNN on Jetson, this is not surprising since CNNs implemented on GPUs have achieved state of the art results in most image classification competitions. However, a Tea network implemented on the NS1e board comfortably outperforms CNNs on the Jetson board when using the $\frac{\|Acc\|}{E_A}$ metric adapted from LPIRC metric. This can be attributed to the high throughput of images on NS1e and the minuscule active power (300 – 400mW) consumed by the NS1e during runtime coupled with a relatively small decrease in accuracy.

VI. CONCLUSION AND FUTURE WORK

We have presented an evaluation of two power-efficient architectures for low-power binary image classification. Our trade study shows that by sacrificing some accuracy, the neurosynaptic TrueNorth processor can significantly improve the accuracy-energy performance as characterized by the $\frac{\|Acc\|}{E_A}$ metric. While the Tegra K1 chip on the Jetson board running CNNs provided the highest accuracies in our study, the TN chip on the NS1e board running Tea networks has demonstrated higher throughput and uses over an order of magnitude less active energy per image classification. We plan to scale this work to more difficult classification and recognition datasets. We also plan to evaluate implementations of CNNs on the TN chip.

REFERENCES

- [1] Arnon Amir, Piyali Datta, William P Risk, Andrew S Cassidy, Jeffrey Kusnitz, Steven K Esser, Alexander Andreopoulos, Theodore M Wong, Myron Flickner, Rodrigo Alvarez-Icaza, et al. Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE, 2013.
- [2] A Cassidy, P Merolla, J Arthur, S Esser, B Jackson, R Alvarez, P Datta, J Sawada, T Wong, V Feldman, A Amir, D Rubin, F Akopyan, E McQuinn, W Risk, , and D. Modha. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013.
- [3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [4] Steve Esser, Rathinakumar Appuswamy, Paul Merolla, John Arthur, and Dharmendra Modha. Backpropagation for energy-efficient neuromorphic computing. In *Advances in Neural Information Processing Systems 28*. NIPS, 2015.
- [5] Steven K Esser, Alexander Andreopoulos, Rathinakumar Appuswamy, Piyali Datta, Davis Barch, Arnon Amir, John Arthur, Alex Cassidy, Myron Flickner, Paul Merolla, et al. Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE, 2013.
- [6] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.
- [7] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia, MM '14*, pages 675–678, New York, NY, USA, 2014. ACM.
- [8] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.
- [9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [11] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.
- [12] Yung-Hsiang Lu, Alan M Kadin, Alexander C Berg, Thomas M Conte, Erik P DeBenedictis, Rachit Garg, Ganesh Gingade, Bichlien Hoang, Yongzhen Huang, Boxun Li, et al. Rebooting computing and low-power image recognition challenge.
- [13] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.
- [14] nVIDIA. Jetson.
- [15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- [16] XILINX. Zynq-7000 all programmable soc overview.
- [17] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.