Dynamic Adaptive Neural Network Arrays: A Neuromorphic Architecture

Catherine D. Schuman
Computational Data Analytics
Oak Ridge National Laboratory
Oak Ridge, TN
schumancd@ornl.gov

Adam Disney
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN
adisney1@vols.utk.edu

John Reynolds
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN
jreyno40@vols.utk.edu

ABSTRACT

Dynamic Adaptive Neural Network Array (DANNA) is a neuromorphic hardware implementation. It differs from most other neuromorphic projects in that it allows for programmability of structure, and it is trained or designed using evolutionary optimization. This paper describes the DANNA structure, how DANNA is trained using evolutionary optimization, and an application of DANNA to a very simple classification task.

CCS Concepts

•Computing methodologies → Neural networks; Genetic algorithms;

1. INTRODUCTION

Neuromorphic hardware is inspired by biological brains. Neuromorphic systems offer a complementary hardware alternative to traditional, von Neumann systems [5]. The major advantage of these systems is their potential ability to intelligently compute with relatively lower power cost. There are many potential applications of neuromorphic computing systems, but perhaps the most important is real-time, *in situ* data processing and analysis. Neuromorphic systems tend to solve the same types of problems as artificial neural networks. Specifically, they tend to emulate spiking neural networks, and thus have the ability to process data that has time components. Neuromorphic hardware is custom-made to represent these networks, so it can emulate larger, more complex neural networks than can be emulated on von Neumann systems, often with much less power.

In a high-performance computing (HPC) setting, neuromorphic hardware has the ability to do a co-processing step on resulting data. Tasks such as classification or anomaly detection can be performed so that points of interest in the data may be flagged for the user as the data is being produced. Thus, rather than storing all of the data and performing analysis later, a neuromorphic system may analyze results from HPC systems in order to pare down the data of interest to the user.

This work describes a neuromorphic hardware effort called Dynamic Adaptive Neural Network Array (DANNA) [2]. It differs from most other neuromorphic projects in that it allows for programmability of structure, and it is trained or designed using evolutionary optimization. This paper describes the DANNA structure, how DANNA is trained using evolutionary optimization, and an application of DANNA to a very simple classification task.

2. RELATED WORK

There are several major efforts to produce neuromorphic computing systems, including IBM's TrueNorth [9], Human Brain Project's SpiNNaker [4] and BrainScaleS [10, 11], and Stanford's Neurogrid [1]. There are also several efforts to build neuromorphic systems using memristors [7, 6, 16]. Neuromorphic computing projects tend to have one of two goals: (1) create hardware that can accurately emulate biological brain behaviors (e.g., BrainScaleS) or (2) create hardware that takes inspiration from biological brains and/or artificial neural networks to attempt to build a useful computing architecture (e.g., IBM's TrueNorth). Some neuromorphic computing projects are striving for both goals simultaneously. Our approach is to work towards the second goal in trying to produce useful and usable computing architectures that are inspired by both biological brain function and artificial neural network constructs.

3. DYNAMIC ADAPTIVE NEURAL NETWORK ARRAYS

Dynamic Adaptive Neural Network Arrays (DANNAs) were inspired by a neural network architecture called Neuroscience-Inspired Dynamic Architecture (NIDA) [13, 12, 14, 15, 3]. NIDA uses spiking neural networks embedded in three-dimensional space that are composed of very simple accumulate-and-fire neurons and synapses with delay and Hebbian learning-type mechanisms. An example NIDA network is shown in Figure 1. NIDA networks are designed using an evolutionary optimization (EO) method, which determines their parameters, structure, and dynamics. We have successfully applied NIDA networks to system control [12], classification [15], and anomaly detection [13] tasks.

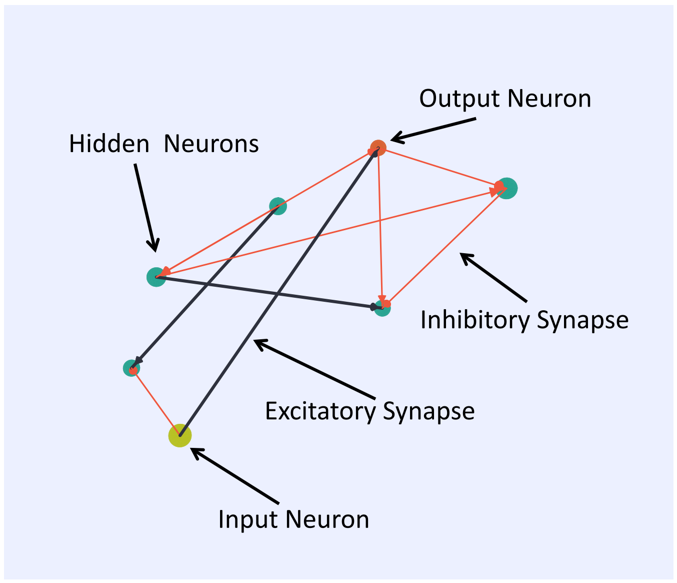

Figure 1: Example NIDA network. NIDA networks are composed of input neurons (yellow), output neurons (red), and hidden neurons (teal), as well as excitatory synapses (navy blue), and inhibitory synapses (red).

DANNA shares many characteristics of NIDA and is composed of accumulate-and-fire neurons and synapses with delay and Hebbian learning mechanisms. DANNA networks are two-dimensional arrays of neuromorphic elements, where each element is programmable as a neuron or a synapse. Each element in the array can connect to up to 16 other elements (its eight immediate neighbors and eight additional neighbors one ring out). Figure 2 shows an example of DANNA network. DANNA has currently been implemented on field programmable gate arrays (FPGAs), and a custom-chip implementation is planned for the near future. The hardware implementation allows for speed-up in simulation of the network; future custom chip implementations will also consume considerably less power than software simulations of the networks.

The primary advantage of DANNAs over other neuromorphic implementations is their programmability. DANNAs are composed of neuromorphic elements, which are programmable as either neurons or synapses, allowing for very flexible network architectures to be defined. This differs

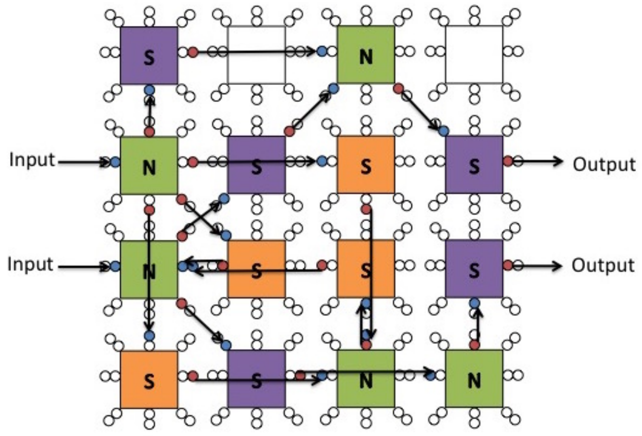

Figure 2: Example DANNA network. Element that are programmed as neurons are green. Excitatory synapses are purple and inhibitory synapses are orange. The circle surrounding each element represent connections to other elements. Blue circles indicate that there is an incoming connection from that neighbor, and red circles indicate that there is an outgoing connection with that neighbor. The black arrows indicate how the elements are connected in this example.

from most other neuromorphic hardware implementations, where the neurons and synapses are fixed (e.g., many use fully-connected networks, which are unnecessary for most applications). This flexibility allows for an optimization method that can determine both the parameters and the structure of the network that is best suited for the task, without changing the underlying hardware.

DANNA networks, such as the one in Figure 2, are designed using evolutionary optimization. This evolutionary optimization design method determines the parameters of the network (such as weights of synapses and thresholds of neurons) and the structure of the network (number and placement of neurons and synapses). This evolutionary optimization method can operate directly on DANNA networks. In most neuromorphic computing efforts, networks are trained using traditional training methods such as back-propagation, and the resulting networks are mapped to the hardware with some loss of performance (as adaptations are made to the network to allow for restrictions in the hardware). A major advantage of evolutionary optimization as a training method is that it can operate directly on the DANNA networks themselves, taking into account both the strengths and the restrictions of the hardware implementation in designing the appropriate network.

DANNA also differs from other neuromorphic systems in the simplicity of the neurons and the synapses. Neurons are defined by their threshold and charge, and synapses are defined by their weight, delay, and refractory period. The weight values of the synapses are also affected by processes inspired by long-term potentiation and long-term depression. These processes cause the weights of synapses to change over the course of simulation as a result of the firing activity in the network.

4. PRELIMINARY RESULTS

We applied our EO to the Iris data set [8]. All of our training was completed using a software simulation of DANNA. The software simulation matches the behavior of the hardware system and has the same interface, so the hardware implementation could easily be used during the optimization. For the Iris data set, there are four input values (petal length and width, and sepal length and width), and the output value corresponds to one of three iris types (Iris Setosa, Iris Virginica, and Iris Versicolour). There are 150 instances in the Iris data set, 50 of each iris type. We split the data into a training set and a testing set, each of which contained 25 instances of each iris type.

For each application, the evolutionary optimization training method requires the user to define the number of inputs, the number of outputs, and a fitness function. In this instance, there are four inputs (one corresponding to each of the input values), and there is one output. Inputs to DANNA networks come in the form of fires on the input neurons, so the input values for petal length and width and sepal length and width have to be converted to fires. To do this, we normalize the input values so that they are integers between 0 and 10, inclusive, and that number of fires is applied to the corresponding input neuron. We simulate activity in the DANNA networks for 500 clock cycles. We then examine the fires on the output synapse for the last 100 clock cycles. Depending on the number of fires in that time window, one of the three iris classes is chosen. The fitness function for this application simulates the network on each of the 75 training instances, and the corresponding fitness value is how many of those 75 instances were correctly classified.

Using a preliminary implementation of the evolutionary optimization for DANNA networks and a 10x10 DANNA network, we produced a network that achieves 96 percent accuracy on the training set and 98.67 percent accuracy on the testing set. This network has 15 neurons and 30 synapses. All of the fitness evaluations were completed using a hardware-accurate software simulation. The resulting network is shown in Figure 3. These results are comparable with NIDA results on the same task. Though the architectures differ slightly because of restrictions in the hardware implementation (such as connectivity restrictions), we may still expect to find DANNA networks that achieve similar performance on tasks to which NIDA networks have been successfully applied.

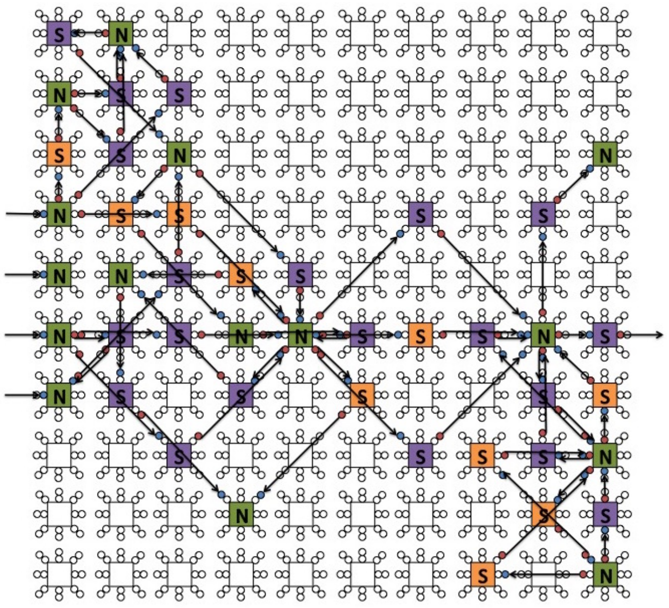

Figure 3: The resulting network for the Iris data set classification task. The same color scheme used in Figure 2 is used here.

5. FUTURE WORK

We intend to pursue several improvements in both software and hardware for DANNA, as well as to tackle more complex applications and tasks. Currently the simulation is clock-based because this approach was the simplest to match the hardware. We intend to explore improvements to the software simulation of DANNA with different algorithmic approaches such as an event queue version as well as parallelizing the simulation using multi-threading, CPU vector instructions (SSE, AVX, etc.), and graphics processing units (GPUs). By improving the software simulation, we will also decrease training time, as we typically use the software simulation during training, due to limited hardware resources.

We intend to explore implementations of the evolutionary optimization design method that is parallelized and scalable. This way, we can take advantage of HPC systems in order to train or design DANNA networks. These networks may then be loaded into the programmable hardware platforms to solve various tasks. Since the evolutionary optimization is reliant on exploration of the search space of solutions, scaling the evolutionary optimization to allow for larger populations will lead to shorter training times. This will allow us to tackle more complex tasks.

We intend to implement development, analysis, and visualization tools to make DANNAs more user-friendly, and we are in the process of developing a software/hardware DANNA kit that will be available to outside users. By making the system accessible to a wider array of users, we hope to more quickly explore and test the limits of this neuromorphic system.

DANNA is currently fully implemented for FPGA. In particular, we have implemented varying sized arrays on various sized FPGAs, with the largest being a 75x75 array of neuromorphic elements. A VLSI implementation and a memristor implementation of DANNA are both in progress. Both of these implementations will allow for increased array sizes, and we also expect that they will lead to reductions in power consumption.

6. CONCLUSION

This work describes a neuromorphic architecture, DANNA, that is trained using evolutionary optimization and is currently implemented on FPGA. There is significant potential for neuromorphic systems to work in companion with existing systems in the future. However, there is still much work to be done in discovering the capabilities of these systems, including which applications they are most appropriate for, and how to efficiently train or design them to complete particular tasks. This work describes preliminary results on a simple classification task using a neuromorphic architecture. We are encouraged by the results, and we intend to apply DANNA to more complicated tasks in order to determine its general applicability to various problem types.

7. ACKNOWLEDGMENTS

This work is in collaboration with the Neuromorphic Computing group at the University of Tennessee. We would like to acknowledge the support and guidance of our collaborators, especially Dr. Mark Dean, Dr. Jim Plank, Dr. Garrett Rose, and Dr. Doug Birdwell.

Prepared in part by Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, Tennessee 37831-6285; managed by UT- Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR2225. This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

8. REFERENCES

[1] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. Merolla, K. Boahen, et al. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014.

[2] M. E. Dean, C. D. Schuman, and J. D. Birdwell. Dynamic adaptive neural network array. In *Unconventional Computation and Natural Computation*, pages 129–141. Springer, 2014.

[3] M. Drouhard, C. D. Schuman, J. D. Birdwell, and M. E. Dean. Visual analytics for neuroscience-inspired dynamic architectures. In *Foundations of Computational Intelligence (FOCI), 2014 IEEE Symposium on*, pages 106–113. IEEE, 2014.

[4] S. B. Furber, D. R. Lester, L. Plana, J. D. Garside, E. Painkras, S. Temple, A. D. Brown, et al. Overview of the spinnaker system architecture. *Computers, IEEE Transactions on*, 62(12):2454–2467, 2013.

[5] G. Indiveri. Neuromorphic engineering. In *Springer Handbook of Computational Intelligence*, pages 715–725. Springer, 2015.

[6] G. Indiveri, R. Legenstein, G. Deligeorgis, T. Prodromakis, et al. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, 24(38):384010, 2013.

[7] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu. Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters*, 10(4):1297–1301, 2010.

[8] M. Lichman. UCI machine learning repository, 2013.

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

[10] T. Pfeil, A. Grübl, S. Jeltsch, E. Müller, P. Müller, M. A. Petrovici, M. Schmuker, D. Brüderle, J. Schemmel, and K. Meier. Six networks on a universal neuromorphic computing substrate. *Frontiers in neuroscience*, 7, 2013.

[11] J. Schemmel, D. Bruderle, A. Grubl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 1947–1950. IEEE, 2010.

[12] C. D. Schuman and J. D. Birdwell. Dynamic artificial neural networks with affective systems. *PloS one*, 8(11):e80455, 2013.

[13] C. D. Schuman and J. D. Birdwell. Variable structure dynamic artificial neural networks. *Biologically Inspired Cognitive Architectures*, 6:126–130, 2013.

[14] C. D. Schuman, J. D. Birdwell, and M. Dean. Neuroscience-inspired dynamic architectures. In *Biomedical Science and Engineering Center Conference (BSEC), 2014 Annual Oak Ridge National Laboratory*, pages 1–4. IEEE, 2014.

[15] C. D. Schuman, J. D. Birdwell, and M. E. Dean. Spatiotemporal classification using neuroscience-inspired dynamic architectures. *Procedia Computer Science*, 41:89–97, 2014.

[16] M. Soltiz, D. Kudithipudi, C. Merkel, G. S. Rose, and R. E. Pino. Memristor-based neural logic blocks for nonlinearly separable functions. *Computers, IEEE Transactions on*, 62(8):1597–1606, 2013.