Short Paper

Neuromorphic Hardware As Database Co-Processors: Potential and Limitations

Visionary

Thomas Heinis[¶], Michael Schmuker[†]

[¶]*Imperial College, London, United Kingdom* [†]*University of Hertfordshire, Hatfield, United Kingdom*

ABSTRACT

Today's databases excel at processing data using simple, mostly arithmetic operators. They are, however, not efficient at processing that includes pattern matching, speech recognition or similar tasks humans can execute quickly and efficiently. One of the few ways to integrate such powerful operators into data processing is to simulate neural networks or to resort to the crowd.

Neuromorphic hardware will become common-place in complementing traditional computing infrastructure and will, for the first time, enable the time-efficient emulation of neural networks. Given the impact hardware accelerators like FPGAs and GPUs had on querying, the question is if neuromorphic devices can be used to a similar effect by enabling richer database operators.

In this paper we thus discuss how neuromorphic devices can be used as database co-processors. The goal of this paper is to understand their potential as well as limitations, what future developments will make them suitable to accelerate databases and what the research challenges are. We also evaluate a prototype implementation of a query operator on neuromorphic hardware.

1 INTRODUCTION

While today computers have a clear advantage over the brain in terms of raw computing power, the brain is superior in error resilience and speed for approximate tasks like understanding speech or recognizing objects. It is for these reasons but also due to the limitations of current CPU designs (pin bottleneck limiting data transfer, limited heat dissipation, power supply) that scientists have developed neuromorphic devices which for the first time enable the energy and time-efficient simulation of spiking neural networks (SNNs).

The vision is that neuromorphic devices will be as prevalent as FPGAs or GPUs are today: in a first instance neuromorphic chips can be plugged into existing systems and computations are offloaded (e.g., image recognition). Later, neuromorphic chips may share sockets and caches to accelerate exchange of data. With Intel, IBM and others developing neuromorphic hardware, the question is not if, but rather when, it will become commonplace.

Research has developed methods by which hardware accelerators are used to accelerate queries. GPUs are used to offload parallel computation (e.g., joins [14]) while FPGAs are used to compute histograms when data is read from storage [13, 20].

The question we thus ask is how neuromorphic hardware will support query execution in databases. In this paper we first discuss what neuromorphic hardware is and how it can be used to simulate spiking neural networks. We then discuss how neuromorphic hardware has the potential to act as a co-processor for databases to offload computation which is best executed using a SNN. We finally also discuss a prototype application.

2 SIMULATING NEURAL NETWORKS

The simulation of neural networks and neuronal activity is not a new development. Perceptron-based multilayer networks have been used for decades in applications like pattern recognition and memory. With neuromorphic hardware, however, it is for the first time possible to efficiently emulate large SNNs.

In neural networks, neurons communicate via spikes, that is, discrete events that occur at defined times. Depending on the weight of the synapse, these events either increase or decrease the probability that the receiving neuron will produce a spike.

Over time, neuron models of increasing bio-realism and thus complexity have been proposed. Early models did not use voltage spikes, but used neurons as threshold gates that simply add input and fire once the threshold is exceeded [18]. These neuron models only take binary input, produce binary output and can thus be used to compute any boolean function.

Next generation neuron models based on the perceptron [24] use an activation function (typically sigmoid or a linear saturation function) on the sum of weighted inputs to compute a continuous, differentiable output. Concatenated ensembles of perceptron units are able to approximate arbitrary functions and can thus be used as universal classifiers in pattern recognition.

More powerful than this are networks with spiking neurons. They use a model of a neuron that receives several spikes, adds them up to the local potential P over time and fires a spike once P exceeds a given threshold. Information can thus not only be encoded in the number of spikes, the firing rate, but also in the timing of spikes [10]. From an information theory point of view we can encode more information and reduce the number of neurons to perform the same computation with this approach.

Additionally, the time-dependence of spiking neural models enables using additional learning algorithms that operate in the time domain. In neural networks that use non-spiking models, learning is achieved through backpropagation of errors [25], where synaptic weights are iteratively adjusted until the applied stimulus leads to the desired output. In spiking networks, this learning rule is extended with spike timing or spike-timing-dependent plasticity (STDP). In STDP, the weight of a synapse changes as a function of the time difference between spikes produced by the pre- and the postsynaptic neuron (neuron before and after the synapse). In STDP, a synapse is strengthened when the pre-synaptic neuron fired a spike shortly before the post-synaptic neuron fired. Vice versa, the synapse is weakened if the post-synaptic spike precedes the pre-synaptic one [2]. STDP thus extends firing-rate based learning by including spike timing.

3 NEUROMORPHIC DEVICES

The brain is a massively parallel system of highly interconnected but computationally simple neurons. Neurons act entirely event driven and thus operate asynchronously, integrating incoming spikes and sending spikes to other neurons.

Integrating spikes is very efficiently done on von Neumann based hardware (traditional CPUs or GPUs) while other aspects cannot be executed efficiently. First, to simulate, a very high number of very small messages (i.e., spikes) must be sent between neurons. While spikes are encoded with a few bits, most

communication protocols require an order of magnitude more bits for header and routing information alone. Moreover, today's communication protocols lack efficient means for multi-cast communication which is crucial to send the same spike to a large set of neurons. Second, while neurons are only active when receiving and processing spikes, traditional CPUs or GPUs are always on, rendering the simulation of SNNs energy inefficient.

3.1 Brain-like Hardware

With these challenges, traditional CPUs or GPUs cannot efficiently simulate SNNs. As a consequence, *neuromorphic hardware* is being developed to better support simulation of SNNs. While the design depends on the manufacturer, all proposals are driven by similar ideas.

Computation: Key to neuromorphic hardware is massive parallelism, i.e., a large number of computational units/cores. Each of the cores has the relatively simple task of simulating several neurons and it can thus be comparatively wimpy (few FLOPS).

Communication: Unlike traditional communication between cores or computers, the payload sent between neurons is very small as only the timing of the spike matters. This information can be encoded in a few bits (40-72). Key to neuromorphic hardware thus is an efficient communication optimized for small payloads.

Given the massive number of connections between neurons, multicast communication must also be efficiently supported.

Energy Efficiency: Since the computational requirements to model neurons are modest, the computing infrastructure can be of low complexity. Also, since neurons operate fully event-driven, there is no need for global synchronization. These properties allow for making neuromorphic devices very energy efficient.

3.2 State of the Art Devices

The prototypes developed share key features like a massively parallel infrastructure with a fast interconnect for small messages.

For example, the SpiNNaker architecture is based on a large number of ARM processors [8]. Memory with little capacity is local to each processor (which itself has several cores) while there is also slower, global memory used for communication between processors. More important is a very efficient communication infrastructure for small packets (~24 Bytes for a spike).

IBM's TrueNorth project is a platform with 4096 cores each simulating 256 neurons [19]. Similar to SpiNNaker, memory, computation and networking is handled locally by each core, therefore moving past the von Neumann architecture. With an event driven model where cores are only powered when needed, TrueNorth reduces energy consumption similarly to SpiNNaker.

Intel's design reduces energy consumption by using spin devices to simulate the neurons (thereby restricting the neuron models that can be used) as well as memristor as local memory (to store synapse weight) [27].

Spikey [23] takes energy efficiency even further by using a mixed-signal approach. Neurons are implemented using analog elements. While this makes them more bio-realistic, it limits the flexibility to use arbitrary neuron models. New models cannot simply be implemented through programming, but require changes in the hardware. Nevertheless, Spikey has proven to be versatile enough to implement a wide range of networks [23].

4 APPLICATIONS IN DATA MANAGEMENT

Spiking neural networks are capable of modelling arbitrary complex processes thanks to their ability to represent different information dimensions, such as time, space, frequency and phase.

In this section we discuss applications where neuromorphic hardware can support databases. In the applications we discuss, extracting all features a priori and storing them in a database may at first seem an option but is unfeasible as the feature space will explode, thus requiring excessive storage space.

The simulation of SNNs is thus key. Any application based on a SNN can be simulated on a CPU or GPU but only inefficiently. The benefit of using neuromorphic hardware is that it can run bigger SNNs faster and crucially, substantially more energy efficient.

While there is a plethora of potential applications which can be simulated with SNNs, our discussion is primarily driven by application domains where SNNs are used successfully.

4.1 Multimedia Databases

Neural networks can be used for data classification [26] but they prove particularly useful for recognising media. Several SNNs have been trained based on supervised learning methods [3, 21, 22, 26] and have been tested on imaging benchmarks. SNNs frequently outperformed non-spiking classification methods [3].

Content-based Image Retrieval Research has produced multiple approaches for content-based image retrieval in multimedia databases [4]. Most approaches are based on two phases: feature extraction (i.e., color, shape and others) and the indexing of the features [17]. Feature extraction is application specific [4] while indexing is general and based on high-dimensional indexes for similarity searches (R-Trees [11] and variants).

State-of-the-art approaches generally have two shortcomings. First, the semantic gap [6] means that it is very challenging to determine the semantics of an image from its low-level features, i.e., there is little connection between pixel statistics and the semantics of an image. Second, due to the high number of dimensions of extracted features and the curse of dimensionality, content-based retrieval of images is not efficient and scales poorly [5].

Spiking neural networks, however, are very useful for image and pattern recognition [1]: using a network with spike-timing dependent plasticity, one of the images is presented to a two layer network in the learning phase. The presentation triggers a single spike in each neuron in the first network layer and the incoming activity propagates to the next layer. The first neuron to fire in the second layer inhibits its neighbors and triggers learning. As a result each neuron fires if again presented with the same image. The learning step is substantially faster compared to other methods (particularly backpropagation) as is the recognition step.

Using spiking neural networks for image retrieval in multimedia databases (or databases in general) thus has two major advantages. First, using SNNs, no feature extraction is necessary and the method consequently can work directly on the raw data. By doing so there is less risk of basing retrieval on a highly specific set of features. Whether the use of SNNs bridges the semantic gap, however, has not been determined yet [1] but it decouples image recognition from features. Second, by using SNNs, no high-dimensional indexes are needed and image retrieval is more efficient and scales better. By simulating SNNs efficiently, neuromorphic hardware enables their use for image retrieval.

Audio & Video Retrieval The problem of audio and video retrieval is very similar to image retrieval with the difference that both also have a time dimension, e.g., videos have consecutive different frames. Simple approaches for video retrieval are based on the same ideas as image retrieval with a additional features drawn from changes between frames. By doing so they have the same drawbacks as current image retrieval methods.

Spiking neural networks support time very well. Any stimuli of a SNN is time-based, e.g., even images need to be encoded as a succession of neuron stimuli over time. SNNs and neuromorphic hardware thus lend themselves well for audio or video [28].

4.2 Spatial Indexing

A vast number of spatial indexes has been developed [9]. Many spatial indexes (particularly based on data-oriented partitioning like the R-Tree and variants), however, suffer from limited performance due to overlap and dead space in the index [11].

SNNs have also been used for spatial navigation: through learning mental maps of the environment it enables planning of paths [12]. The particular neuron in the network is activated as a simulated animal explores different locations in the environment and connections between neurons activated in a close temporal proximity are strengthened, i.e, cells representing neighboring locations develop strong synaptic interactions. This mechanism can be used for answering nearest neighbor queries.

A SNN run on neuromorphic hardware thus has the potential to execute nearest neighbor queries and planning paths.

5 CHALLENGES

We propose to use neuromorphic hardware as a co-processor for databases. We use the hardware to efficiently simulate a SNN for a complex query operator, e.g., pattern recognition for image retrieval. The associated research challenges can be identified for neuroscience and data management research alike.

On the level of hardware, while existing neuromorphic hardware already scales substantially better than traditional von Neumann architectures, it has to be investigated how current approaches can scale to simulate SNNs beyond millions of neurons, so that more sophisticated operators can be implemented.

Regarding neuroscience, the challenge lies in finding more SNNs that solve generic computing problems and are amenable to simulation on neuromorphic hardware. Several such SNNs have been developed but more applications from human cognition need to be considered as they are useful as database operators.

On the level of databases the challenges are as follows:

5.1 Data Preparation/Encoding

The data in the database and the query need to be encoded for the neuromorphic hardware. More precisely, as is common for SNNs, the data needs to be encoded as temporal spikes that can be fed into the system. Consider, for example, pattern recognition in images. All images as well as the query need to be encoded as temporal spiking patterns for it to be matched on a SNN.

Encodings for video and audio can be computed very efficiently: because they already have a temporal structure, data like movies or sound are converted to sequences of spikes using little computational effort. Similarly, for imaging data, sensors are available that efficiently produce time series of spikes [16].

As numerous examples of functional SNNs show, it is often possible to convert arbitrary data into spikes. Whether the effort to produce such an encoding is outweighed by the gain in computational power that a SNN will provide over more conventional approaches of data processing, however, is questionable.

Furthermore, while the same encoding can be used for matching different queries, it is very likely that different query operators will benefit from encoding that is tuned to that particular operator. A more difficult challenge thus is to decide what encodings should be stored and the organization in storage. It may suffice to only store the difference between encodings instead of storing one encoding per operator.

The most difficult challenge, however, is how to execute queries on multiple attributes. Going back to the imaging example: a query may restrict the area in which to find a particular pattern. Restricting the area on an already encoded imaging is difficult as all positional information (about the area) is lost in the encoding.

5.2 Query Planning

Current prototypes of neuromorphic hardware have a very limited interface to move data making query execution challenging.

Modelling Execution Cost A crucial question is when SNNs are run faster on neuromorphic hardware compared to the CPU. The query plan has to consider a cost model with the time to transfer data to and from the device and the execution time of the simulation. Execution time, however, is not the only consideration as running an operator on the device is more energy efficient than on the CPU. A second consideration thus has to be the energy needed to move the data and to run the SNN.

Query Optimization Given the high cost of moving data, moving all data to the device should be avoided. Considering the selectivity of predicates other than SNN (e.g., metadata information) early in the query plan is therefore imperative. Further, a crucial research challenge is to investigate if features in the raw data can be identified, extracted and indexed such that they can be used early in the query execution to curb data movement. Selecting the features is very challenging without reintroducing the semantic gap known from image retrieval.

5.3 Query Execution

Setting Up New Operators To define an operator, a network needs to be trained by adjusting synaptic weights. Key to the training phase is to encode the stimulus appropriately and injecting all learning stimulus consecutively to train the network. The learning phase can also be performed in a simulated environment and the resulting weights can be transferred to the hardware.

Swapping Operators One network cannot serve all purposes so operators/networks need to be loaded and unloaded. The state of a SNN or operator is captured in the neurons, their placement, their interconnections and the synaptic weights.

To swap an operator, a new SNN needs to be set up by loading neurons and synapse weights which is a slow process. In many cases, however, it may suffice to only define rules and derive the precise connections/network on the fly.

Interpreting Results Depending on the SNN, the result is indicated differently. In some cases the result is binary, i.e., one neuron being active indicates the result. In other cases, for example in case of the spatial application, the proximity of two points is expressed by the proximity of two active neurons. Each operator must therefore come with an implementation of an interpreter able to understand the result.

The major research challenge is how ambiguous simulation results are interpreted. The result is rarely precise and this ambiguity has to be propagated to the user. One approach is the use of ideas from uncertain databases, e.g., attribute-level uncertainty.

6 CURRENT LIMITATIONS

Neuromorphic hardware still is a new proposition and most available prototypes focus on the efficient execution of SNNs as a proof of concept whereas the programmability, data transfer and others are currently only second order considerations.

Moving Data: Current hardware, in particular the SpiNNaker system, has a very limited interface to move data to and from the device (Ethernet interface with 100MBps). The issue of bandwidth will, however, be addressed if neuromorphic hardware demonstrates its usefulness.

Simulation Size: The current challenge for hardware developers is increasing the capacity of neuromorphic systems. The largest systems today can simulate at most millions of neurons. Nevertheless, it is believed that scaling the number of neurons up to the human brain (i.e., 10^{10} neurons) will give neuromorphic systems the ability to infer relationships in data of a complexity that is inaccessible to conventional computers [7].

7 PROTOTYPE IMPLEMENTATION

To show the basic feasibility of using neuromorphic hardware as a database co-processor, we implement a proof of concept based on a simple application. Clearly this is only an example with several obvious optimizations which we address in future work.

7.1 Sample Application

As our sample application we use the recognition of digits in images [26]. Images containing hand written digits are stored as bitmaps in the database and a users query to find images containing a particular digit. To answer a query, a SNN run on neuromorphic hardware infers the digit in the image. The SNN is trained offline by showing sample digits (0 - 9) from MNIST [15].

In the current implementation we use the same digits for learning and querying. We do so to avoid the challenge of interpreting the result — an open research challenge as we discussed. Hence, when training we store the response (active neurons) to the stimulus for a particular digit. When asking what images contain a particular digit, we present the stored images to the SNN and compare the stimulus response (active neurons) with stored responses. Due to the nondeterminism of SNNs, the stored and the current result may not be identical. We thus compute the share of active neurons the current and the stored response agree on and consider the result a match if it exceeds a predefined threshold.

Compared to preprocessing all data and storing the result, we remain flexible as the SNN can be updated and ran on the hardware to query the data (with potentially more accurate results).

7.2 Setup

For the current implementation we use Postgres. Given the images from MNIST are stored as bitmaps, we store them as two dimensional arrays without need for transformation.

We use a user defined C function in Postgres: given a number n and a images stored in a table t, the UDF presents all images in t to the classifier and returns the ones likely to contain n.

For the experiments we use a 4 chip SpiNNaker board [8], the smallest board but adequate for a proof of concept. The board has 72 identical Arm968 processors (18 per SpiNNaker chip) operating at 200MHz. The SpiNNaker board currently only provides an Ethernet interface which connect to the host running Postgres.

7.3 Experimental Analysis

Training the SNN takes on average 68.2 seconds. The vast majority (95.5%) of time is spent transferring data: images to the device (8.4%), reading the stimulus responses (29.2%) and reading the trained SNN (57.9%). Learning only takes 3.1 seconds as each of the hundred digits is exposed to the SNN for 20ms.

Querying for one number — classifying all images and finding the ones matching the user input — takes on average 52.5 seconds. Most time is spent on moving data, i.e., moving the trained SNN (84.4%) and sending all images (10.5%) to the device. Classification on the device takes 0.2s while the time spent in the UDF is 2.48s.

Clearly, querying (as well as learning) suffers from loading (and storing) the SNN — the main bottleneck. Even without, however, moving the images is a very costly operation. The time for learning and classifying, on the other hand, is insignificant.

7.4 Open Challenges

As simple as the example used is, it shows some of the challenges discussed. A major challenge is the transfer of data. Currently this can only be done through Ethernet and is slow but future versions will have SATA and TrueNorth, for example, uses PCI.

Related to the limited bandwidth is the challenge of reducing the data moved: only images which contain the digit with high probability should be moved. Means to index the stimuli need to be devised so that a preselection of images can be performed.

Finally, scoring the result (and the inherent uncertainty) is a challenge which we worked around. Clearly, better approaches need to be devised for more sophisticated applications.

8 CONCLUSIONS

While neuromorphic hardware systems are still maturing, the high-level design is well-defined and the benefits are clear. It is thus the right time to start assessing the viability of neuromorphic applications. In this paper we have sketched the research challenges for query execution. Some challenges are implied by the prototypic nature of the hardware, but most are due to more fundamental reasons (e.g., query planning). By addressing these challenges we believe neuromorphic hardware will enable the efficient execution of more sophisticated query operators.

REFERENCES

[1] 2001. Spike-based Strategies for Rapid Processing. *Neural Networks* 14, 67 (2001), 715 – 725.

[2] G Bi and M Poo. 2001. Synaptic Modification by Correlated Activity: Hebb's Postulate Revisited. *Annual Review of Neuroscience* 24 (Jan 2001), 139–166. https://doi.org/10.1146/annurev.neuro.24.1.139

[3] S. M. Bohte, H. La Poutre, and J. N. Kok. . Unsupervised Clustering with Spiking Neurons by Sparse Temporal Coding and Multilayer RBF Networks. *Transactions on Neural Networks* 13, 2 ().

[4] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. 2008. Image Retrieval: Ideas, Influences, and Trends of the New Age. *Comput. Surveys* 40, 2 (May 2008).

[5] Adrien Depeursinge, Benedikt Fischer, Henning Müller, and Thomas M Deserno. 2011. Prototypes for Content-Based Image Retrieval in Clinical Practice. *The Open Medical Informatics Journal* 5 (Jul 2011).

[6] Thomas M. Deserno, Sameer Antani, and Rodney Long. 2008. Ontology of Gaps in Content-Based Image Retrieval. *Journal of Digital Imaging* 22, 2 (2008).

[7] Steven K. Esser, Alexander Andreopoulos, Rathinakumar Appuswamy, Pallab Datta, Davis Barch, Arnon Amir, John Arthur, Andrew Cassidy, Myron Flickner, and Paul Merolla. 2013. Cognitive Computing Systems: Algorithms and Applications for Networks of Neurosynaptic Cores. December (2013).

[8] S.B. Furber, F. Galluppi, and S. Temple. 2014. The SpiNNaker Project. *Proc. IEEE* 102, 5 (2014).

[9] Volker Gaede and Oliver Guenther. 1998. Multidimensional Access Methods. *Comput. Surveys* 30, 2 (1998).

[10] Wulfram Gerstner, Andreas K. Kreiter, Henry Markram, and Andreas V. M. Herz. 1997. Neural Codes: Firing Rates and Beyond. *Proceedings of the National Academy of Sciences* 94, 24 (1997).

[11] Antonin Guttman. . R-trees: a Dynamic Index Structure for Spatial Searching. *SIGMOD* ().

[12] John J. Hopfield. 2010. Neurodynamics of Mental Exploration. *Proceedings of the National Academy of Sciences* 107, 4 (2010), 1648–1653.

[13] Zsolt Istvan, Louis Woods, and Gustavo Alonso. . Histograms As a Side Effect of Data Movement for Big Data. In *SIGMOD '14*.

[14] Tim Kaldewey, Guy Lohman, Rene Mueller, and Peter Volk. . GPU Join Processing Revisited. In *Proceedings of the Eighth International Workshop on Data Management on New Hardware (DaMoN '12)*.

[15] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 1998. The MNIST Database of Handwritten Digits.

[16] P. Lichtsteiner, C. Posch, and T. Delbruck. . A 128 X 128 120db 30mw Asynchronous Vision Sensor that Responds to Relative Intensity Change. In *IEEE International Solid-State Circuits Conference 2006*.

[17] M. De Marsicoi and L. Cinque. 1997. Indexing Pictorial Documents by their Content: A Survey of Current Techniques. *Image and Vision Computing* 15, 2 (1997).

[18] WarrenS. McCulloch and Walter Pitts. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics* 5, 4 (1943), 115–133. https://doi.org/10.1007/BF02478259

[19] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, and Nabil Imam. 2014. A Million Spiking-neuron Integrated Circuit with a Scalable Communication Network and Interface. *Science* 345, 6197 (2014).

[20] Rene Mueller and Jens Teubner. . FPGA: What's in it for a Database?. In *SIGMOD '09*.

[21] Emre Neftci, Srinjoy Das, Bruno Pedroni, Kenneth Kreutz-Delgado, and Gert Cauwenberghs. 2014. Event-driven Contrastive Divergence for Spiking Neuromorphic Systems. *Frontiers in Neuroscience* 7, January (2014).

[22] Bernhard Nessler, Michael Pfeiffer, Lars Buesing, and Wolfgang Maass. 2013. Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity. *PLoS Computational Biology* 9, 4 (2013).

[23] Thomas Pfeil, Andreas Grübl, Sebastian Jeltsch, Eric Müller, Paul Müller, Mihai A. Petrovici, Michael Schmuker, Daniel Brüderle, Johannes Schemmel, and Karlheinz Meier. 2013. Six Networks on a Universal Neuromorphic Computing Substrate. *Frontiers in Neuroscience* 7 (2013), 11.

[24] Frank Rosenblatt. 1958. The perceptron: a Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* 65, 6 (1958).

[25] Rumelhart, David E and Hinton, Geoffrey E. 1988. Learning Representations by Back-propagating Errors. *Cognitive Modeling* 5 (1988), 3.

[26] Michael Schmuker, Thomas Pfeil, and Martin Paul Nawrot. 2014. A Neuromorphic Network for Generic Multivariate Data Classification. *PNAS* 111, 6 (Feb 2014).

[27] Mrigank Sharad, Charles Augustine, Georgios Panagopoulos, and Kaushik Roy. 2012. Proposal For Neuromorphic Hardware Using Spin Devices. arXiv:arXiv:1206.3227

[28] Simei Gomes Wysoski, Lubica Benuskova, and Nikola Kasabov. 2008. Fast and Adaptive Network of Spiking Neurons for Multi-view Visual Pattern Recognition. *Neurocomputing* 71, 13 - 15 (2008).