A Comparison of Neuromorphic Classification Tasks

John J. M. Reynolds
James S. Plank
Catherine D. Schuman
Grant Bruer
Adam W. Disney
Mark Dean
Garrett S. Rose

July, 2018

International Conference on Neuromorphic Systems (ICONS 2018)

https://ornlcda.github.io/icons2018/

Copyright ACM. Please find the ACM version of the final camera-ready paper at http://dl.acm.org/citation.cfm?id=3229884

The online home for this paper may be found at: http://neuromorphic.eecs.utk.edu

Citation – Plain Text:

.inproceedings{rps:18:acn,
  title = "A Comparison of Neuromorphic Classification Tasks",
  booktitle = "International Conference on Neuromorphic Computing Systems",
  publisher = "ACM",
  address = "Knoxville, TN",
  year = "2018",
  doi = "10.1145/3229884.3229896",
  where = "https://dl.acm.org/citation.cfm?id=3229896"
}

Bibtex:

@INPROCEEDINGS{rps:18:acn,
  title = "A Comparison of Neuromorphic Classification Tasks",
  booktitle = "International Conference on Neuromorphic Computing Systems",
  publisher = "ACM",
  address = "Knoxville, TN",
  year = "2018",
  doi = "10.1145/3229884.3229896",
  where = "https://dl.acm.org/citation.cfm?id=3229896"}
A Comparison of Neuromorphic Classification Tasks

John J. M. Reynolds, James S. Plank
University of Tennessee
Electrical Engineering & Computer Science
[jreyno40,jplank]@utk.edu

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

Grant R. Bruer, Adam W. Disney, Mark E. Dean, Garrett S. Rose
University of Tennessee
Electrical Engineering & Computer Science
[gbruer,adisney1,markdean,garose]@utk.edu

ABSTRACT

A variety of neural network models and machine learning techniques have arisen over the past decade, and their successes with image classification have been stunning. With other classification tasks, selecting and configuring a neural network solution is not straightforward. In this paper, we evaluate and compare a variety of neural network models, trained by a variety of machine learning techniques, on a variety of classification tasks. While Deep Learning typically exhibits the best classification accuracy, we note the promise of Reservoir Computing, and evolutionary optimization on spiking neural networks. In many cases, these technologies perform as well as, or better than Deep Learning, and the resulting networks are much smaller than their Deep Learning counterparts.

KEYWORDS

Neuromorphic, Spiking Neural Network, Deep Learning, Reservoir Computing

1 INTRODUCTION

Neural networks and machine learning are ubiquitous in today’s computing landscape. Given the incredible successes of Deep Learning approaches to image classification, any scientist or researcher, when confronted with a task that involves the classification of data, is compelled to try some sort of neuromorphic machine learning approach to solve his or her problem. The researcher quickly discovers, however, that applying the successes of current neuromorphic approaches to problems beyond image classification is challenging, and there is very little credible guidance. As such, researchers typically attempt to convert their problems to image classification, and then apply one of the myriad image classification Deep Learning models to solve it [11]. Or, they use techniques known as “transfer learning” or “fine-tuning,” where they apply and then refine a Deep Learning solution (that was originally developed for image classification) to their data in hopes that it will work.

Besides Deep Learning, there are other neuromorphic approaches to data classification that hold promise because they rely on highly recurrent neural networks, rather than the feed-forward neural networks employed by Deep Learning, to attack the problem. Two notable examples are Reservoir Computing (RC), where a fixed and highly recurrent “reservoir” is employed as a pre-processing step to convert complex data into data that is linearly separable, and Evolutionary Optimization of Neuromorphic Systems (EONS), where highly recurrent spiking neuromorphic systems are trained via genetic algorithms to solve specific problems.

In this paper, we embody a researcher who desires to use neuromorphic machine learning to solve some data classification problems, and we evaluate six approaches that such a researcher might consider. Four of them are Deep Learning variants, available in the Keras Deep Learning framework. The other two employ RC and EONS to leverage a spiking neuromorphic system called NIDA, to classify data. In each approach, we do not attempt to refine the approach to each specific problem, as we are embodying a scientific researcher who desires to use neuromorphic machine learning as a tool, rather than a computer scientist researching neuromorphic computing. We found that, in general, there was not one machine learning approach that performed best overall. However, both the spiking neural network (RC and EONS) approaches resulted in much smaller network sizes than the Deep Learning approaches, and thus can result in more size and power efficient neuromorphic implementations.

2 NEURAL NETWORKS AND MACHINE LEARNING ALGORITHMS

The general approach of neuromorphic machine learning is similar across all of the techniques that we evaluate in this paper. We are given a collection of data points that are partitioned into different
classes. We use this data to train the components of a neuromorphic system (e.g., the weights of the synapses), so that the system may be employed to predict the classes of subsequent data. The components that get trained, and the techniques for training, differ among the systems that we evaluate. We explain these in the following subsections.

2.1 Deep Learning Approaches

Deep Learning is a term that encompasses layered, (mostly) feed-forward neural networks that are trained with back-propagation. At a high level, the neurons in the system hold values, and synapses propagate values from one neuron to another. Synapses have weights, and each neuron’s value is calculated as a function of its incoming synapses’ weights and propagated values. The neurons are partitioned into layers, and the connectivity from layer to layer is predefined. Dense layers have full connectivity from neurons in the previous layers, while other layers such as convolutional or maxpooling layers have more restricted connectivity.

A model in a Deep Learning system is the definition of the layers, their connectivity, and the mathematical functions that compute neuron values from their incoming synapses. Once a model is defined, the training process uses back-propagation to define the weights of the synapses. Back-propagation uses a gradient descent-style process to iteratively reduce the error of the system when applied to the training data. If a model is more complex, the training process will be slower; as such, many researchers make their models, and often the training weights, public so that other researchers may benefit from their training process.

There are a variety of open-source Deep Learning systems such as TensorFlow, CNTK, and Theano [1, 28, 32]. The Keras Deep Learning framework is a high-level framework and API in Python, that allows one to design and employ Deep Learning models with any of these as their back-end [5]. We have developed a very simple data classification framework in Keras that allows us to employ multiple models to classify data, and in this paper, we evaluate four models:

- **Perceptron** is a simple model with just two layers of neurons: the inputs and the outputs. The inputs are fully connected by synapses. As such, this model can perform linear separation of its data. The output is passed through a softmax activation function to generate a probability distribution for classification purposes.

- **Multi-Layer Perceptron** is composed of three fully connected layers, each containing 64 neurons, employing the ReLU activation function, with a softmax at the end. Sometimes termed “vanilla neural networks,” Multi-Layer Perceptron networks are known to be good classifiers, with the multiple layers and non-linear activation functions allowing them richer functionality than linear classifiers [13].

- **Conv** “LeNet-5” is a well-known convolutional model first developed for handwriting recognition [16]. It is easily modifiable for different input structures, and therefore is our archetypical model for convolutional Deep Learning networks.

- **LSTM**: “Long Short-Term Memory” is an enhancement to the neuron structure of standard feed-forward neural networks to add some recurrence, and ideally some memory, to the networks [14]. Stacking LSTM layers is a commonly used technique in order to better learn higher dimensional features and time-series data [12]. Our LSTM implementation contains three layers, each with 32 cells.

2.2 Spiking Recurrent Neural Networks

Spiking neural networks work very differently from the Deep Learning neural networks described above. In spiking neural networks, there is an explicit time component, and rather than holding values, neurons hold charge. Synapses transmit charge over time, and when a synapse’s charge arrives at a neuron, it is added to the neuron’s charge value. If this neuron’s charge exceeds a threshold, the neuron fires, resetting its charge value to some base value and sending charge out along its outgoing synapses.

In a spiking neural network, neuron thresholds, synapse weights and synapse delays are configurable. When the networks are recurrent, there is no constrained structure as there is with Deep Learning networks. The goal of training a spiking recurrent neural network is to define the connections, thresholds, weights and delays so that the network can “solve” an application. With a classification application, the values of the data points must be converted into spikes, and output spikes must be converted into classifications. We will describe how that works in Section 2.3 below.

The spiking neural network system that we use is called Neuroscience Inspired Dynamic Architecture (NIDA) [24]. This is a very simple system featuring neurons configured in three dimensional space and unlimited connectivity between neurons. Weights and thresholds are floating point values, and delays correspond to the Euclidean distance between neurons. NIDA has a few additional features such as refractory periods for neurons, where they may accumulate charge without firing, and long term potentiation/depression, (LTD/LTP), where synapse weights grow and shrink according to how responsible they are to their target neuron’s firing. NIDA has demonstrated success at applications such as data classification and control [23, 25, 27].

2.3 EONS: Evolutionary Optimization of Neuromorphic Systems

Spiking recurrent neural networks like NIDA are not static and tightly constrained as feed-forward neural networks are. Thus, they cannot be easily programmed by back-propagation, and other techniques must be employed to make them work effectively. EONS [27] takes a genetic algorithm approach to training networks. In particular, the following steps are taken in EONS to generate a spiking recurrent neural network for a particular application:

1. An initial population of networks is generated randomly. Heuristics may be employed to intelligently initialize networks, for example, to force input neurons to have paths to output neurons.

2. Each network in the population is evaluated and given a fitness value. This is done by having the application apply a training suite of tasks to it (e.g., sweeping through a training
set of data) and measure its success (e.g., calculating the accuracy of the classification).

3. The members of the population with the highest fitness are selected for reproduction, which involves mutating parameters of single networks, and performing crossover operations on pairs of networks.

4. Return to step two with the population composed of these newly generated networks.

5. Quit when the fitness achieves a desired threshold, or after a specified period of time has passed.

EONS is advantageous because of its generality — so long as an application can define its fitness suite and a neuromorphic system can define the reproduction operations, then EONS can train networks. Its major problem is that its search space can be colossal, and therefore it can be slow (or unable) to converge. EONS has been applied successfully to various control and classification problems on multiple neuromorphic systems [19, 22, 26, 27].

For EONS, we use the TENN-Lab Hardware/Software Co-Design Framework [22], which provides application and device support for spiking neuromorphic systems. This framework includes a general classification application, which trains networks for given labeled classification data sets. The application assumes that the each data point is composed of \((r \times c)\) values. It then employs a network with \(r\) input neurons and inputs the values in \(c\) groups, one per neuron. The groups are separated by 5 units of simulated time. The values themselves are converted into spikes normalized from -1 to 1.

There is one output neuron per class. For each data set, the network runs for \(N\) time steps from when the first wave of values begins, \(N\) being chosen based on the application. At the end of the \(N\) time steps, the output neuron that fires the most determines the classification. The TENN-Lab framework supports multiple neuromorphic computing architectures, one of which is NIDA.

### 2.4 Reservoir Computing

Reservoir Computing [15, 18] is a paradigm that attempts to harness the power of a highly recurrent neural network (“the reservoir”) by training a “readout” layer to interpret its output. A high level diagram is in Figure 1. In this work, the reservoir is a large NIDA network, generated randomly. It processes its data for classification like the EONS application, with \(r\) input neurons for \((r \times c)\) input values. Each network has 100 output neurons. During the spiking neural network simulation, firings are counted for the last \(M\) time steps, with \(M\) being chosen based on the application. These counts compose the state vector of Figure 1, which is then input to the perceptron layer. The outputs of the perceptron layer are then passed to a softmax layer, as in the Deep Learning network. The reservoir in this system does not change once it is created. For NIDA, this means that the LTD/LTP synaptic plasticity process is turned off. Only the weights of the perceptron layer are trained using back-propagation.

![Figure 1](image)

**Figure 1:** This is a simple diagram of the reservoir computing paradigm. In our work, the reservoir is made up of a recurrent spiking neural network model known as NIDA. The single layer connected to the weight matrix is the output layer.

Most research on RC is heavily laden with mathematics. One body of work employs non-spiking networks for the reservoir. These are called Echo State Networks [15]. A second body of work employs recurrent, spiking networks like NIDA. These are called Liquid State Machines [18]. A third body of work employs single neurons and delayed feedback loops as their reservoirs. These are called Time Delay Reservoirs [4]. With all three models, “good” reservoirs are defined by properties such as, for example, input separability and fading memory for Liquid State Machines [18]. Many works explore optimizing based on these parameters [9], but that expands beyond the thesis of this paper.

### 2.5 Summary of Neural Networks and Machine Learning Algorithms

Table 1 summarizes the neural networks and machine learning algorithms that we employ in this paper. We note that while Keras and TensorFlow are both open-source software packages, at present the TENN-Lab software framework is not. The TENN-Lab authors welcome collaborations with application and neuromorphic architecture teams who wish to explore their software.

<table>
<thead>
<tr>
<th>Name</th>
<th>Software</th>
<th>Neural Network</th>
<th>Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>Keras / TensorFlow</td>
<td>Feed Forward</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>MLP</td>
<td>Keras / TensorFlow</td>
<td>Feed Forward</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>Conv</td>
<td>Keras / TensorFlow</td>
<td>Feed Forward</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>LSTM</td>
<td>Keras / TensorFlow</td>
<td>Nodes with recurrence</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>EONS</td>
<td>TENN-Lab / NIDA</td>
<td>Spiking, Recurrent</td>
<td>Evolutionary Optimization</td>
</tr>
<tr>
<td>Reservoir</td>
<td>TENN-Lab / NIDA</td>
<td>Spiking, Recurrent + Perceptron</td>
<td>Random + Back-Propagation</td>
</tr>
</tbody>
</table>

Table 1: Summary of the neural networks and machine learning algorithms employed in this paper.

### 3 CLASSIFICATION APPLICATIONS

This section introduces and briefly describes the classification data sets that we explore in this work. The data sets come from a variety of sources and have different properties that make them interesting.
3.1 Iris, Breast Cancer, Pima
These are three well-known static classification datasets, available from the UCI Machine Learning Repository [17]. By static, we mean that there is no temporal component to this data. For EONS this means that $c$ equals one. This simplification required us to change a few of the neural networks models. For Convolutional, we simplify the LeNet-5 model from a 2D convolutional model to a 1D model. For RC, because Reservoir Computing relies on dynamics generated within the reservoir, the features were mapped into multiple pulses over time for continuous stimulation. This is necessary for high accuracy classification of static tasks in reservoir computing [2].

The Iris data set is quite old [10] and is very frequently used as a first classification task for machine learning experiments. It consists of 50 data points for each of three classes of flowers: Setosa, Virginica, and Versicolor. Each data point has four values: the length and width of both the sepals and the petals. The Wisconsin Breast Cancer (WBC) dataset is composed of data collected from digitized images of a fine needle aspirate of a breast mass [30]. There are 699 data points, composed of 10 data values per data point. The classifications are whether the mass is benign or malignant. The Pima Indians Diabetes dataset is a set of 768 multivariate samples from patients that either have diabetes or do not. The goal is to determine whether a given patient exhibits the symptoms based on the number of pregnancies, glucose concentration, blood pressure, skin thickness, insulin levels, body mass index, pedigree function, and age.

3.2 Satellite Radio
The Radio [dataset published by DeepSig, Inc. [8], and contains 128-sample snippets of complex-valued temporal radio signals that have been modulated according to 11 different waveforms and 20 different signal-to-noise ratios. Each waveform contains the same number of snippets per signal-to-noise ratio, and there are between 4,200 and 24,940 snippets per waveform. The curators of this data have applied a custom 5-level Deep Learning network for classifying this data, and achieved roughly 95% accuracy when focusing on data with the highest signal-to-noise ratio [21]. For our testing, we selected one modulation type (8PSK) and tested classification performance of that type vs. all of the others, with the highest signal to noise ratio. We trained on 667 snippets, equally divided between 8PSK and the others, and tested on 166 different snippets, also equally divided. We chose the 8PSK modulation type due to its mis-classification with other signals in [21], making it good for individual comparison. We show two example snippets from this data set in Figure 3.

3.3 Electroencephalogram (EEG)
The Electroencephalogram (EEG) is a measure of activation signals in the brain over time. An EEG can contain lots of interesting information within the signal. Because of this, statistical measures have been used to extract relevant and useful metrics. Many works explore onset detection of epilepsy from EEG signals [29]. However, in this work, we perform standard classification rather than onset detection. The dataset used for classification is made publicly available by Andrzejak [3].

The dataset has five sets of EEG signals labeled $A$ through $E$, composed of 23.6 seconds of signals recorded with a sampling rate of 173.61 Hz, for a total of 4,097 samples per data set. For classification, we employed 200 sets of signals split equally between healthy (from set $A$) and epileptic (from set $E$). We display two of these data sets in Figure 2. To be consistent with the literature, we used 80 signals from each set for training, and 20 for testing [31].

3.4 Consonants vs. Vowels
The TIMIT dataset contains audio files of ten sentences spoken by each of 630 speakers, sampled at 16 kHz. We convert each WAV file into Mel-frequency Cepstral Coefficients (MFCCs): vectors of
Table 2: Summary of classification data sets and their parameters as they affect the various machine learning algorithms. \( N \), which corresponds approximately to spiking neural network run-time, applies only to EONS and RC, and \( M \), which corresponds to when output pulses are counted, applies only to RC.

12 coefficients, where each vector represents a slice of the spoken sentence [7]. The MFCCs are computed every 10 ms, over a sample size of 25 ms. To the 12 coefficients, we add a 13th value, the log energy [33]. An example phoneme spectrogram is included in Figure 4. Being able to discern between vowels and consonants has important implications for speech recognition [6]. As such, we have partitioned the data set into two classifications: vowels and consonants. To be consistent with the work by Norton and Ventura [20], we employed a training set of 2000 values, equally split between consonants and vowels, and a testing set of 200 values.

![Spectrogram of MFCC features from a spoken phoneme 'iy'; a vowel. Each column is a MFCC vector with 13 values, calculated every 10 ms over 25 ms of samples.](image)

**Figure 4**: Spectrogram of MFCC features from a spoken phoneme 'iy'; a vowel. Each column is a MFCC vector with 13 values, calculated every 10 ms over 25 ms of samples.

### 3.5 Summary of Data Sets

Table 2 summarizes the data sets that we explore in our experiments. For each data set, we summarize the \((r \times c)\) values for each data point that is input to the neural networks, the number of classifications, the partitioning of data points into training and testing, the number of time steps for the spiking neuromorphic networks, and the output window over which spikes are counted for the RC test.

### 4 EXPERIMENTAL SETUP

The Deep Learning tests were executed on an Intel Core i7-7700 CPU, running at 3.60 GHz with 16 GB of RAM. Because EONS and RC are more computationally demanding during training, we performed them on a 44-node, 1772-core cluster of AMD Opteron CPUs running at 2.3 GHz. Eight of the nodes have 24 GB of RAM, and the remaining nodes have 96 GB of RAM.

With the Deep Learning models, the \((r \times c)\) data points were flattened and applied into the networks simultaneously; with EONS and RC, they were input in \(c\) waves of \(r\) points, as described in Section 2.3. For Deep Learning, we ran 10 training runs of each method, each with a different random number seed, and trained for 1024 epochs. For EONS, we ran 100 training runs, each with a population size of 100 networks, for one hour each on a single CPU core. For RC, we generated 100 random reservoirs, and then trained the readout layer for 1024 epochs.

### 5 RESULTS

The classification accuracies on the testing data of the best network for each model are shown in Table 3 and Figure 5. With the exception of EEG, for which there is no comparison, the best classification accuracy for each data set is on par with, or exceeds previously published data [20, 21, 27].

![Table 3: Percentage accuracies of each model on each data set. Each number presents the best classification accuracy on the testing set of data.](image)

Perhaps the most important thing to note from the results is that there is no consistent "best method" over all of the data. EONS displays the best classification accuracy for Iris, WBC and EEG (tied with Conv); the Convolutional Deep Learning method displays the best accuracy for Radio, EEG and TIMIT; and LSTM and Reservoir tie for the best at Pima.

With the exception of Iris, on which all of the models perform well, the Perceptron model classifies significantly worse than the other models. This is unsurprising, since the Perceptron model has no hidden layer, and thus restricted computational abilities.
The Multi-Layer Perceptron (MLP) improves accuracy significantly over the single Perceptron with the WBC, Pima and TIMIT data sets. The Conv model performs poorly only on the Pima data set, and it performs better than all of the other models on the Radio data set. As described by O’Shea et al, this is because the Radio data set exhibits features similar to image classification [21]. It is surprising that LSTM performs poorly on the Radio and EEG data sets, given that they are time-series data sets, for which LSTM was developed. We surmise that this is due to two separate reasons. For EEG, the number of samples (4097) is still considered to be a very long sequence, and the LSTM paradigm is impacted by the vanishing gradient problem. By exploring parameter settings in Keras and breaking the sequence into subsequences, LSTM could improve. Since a goal of this paper was to inhabit the perspective of a scientist wanting to classify data, and not a computer scientist exploring Deep Learning, we intentionally did not spend time exploring parameter settings.

The spiking neuromorphic models perform well with two exceptions. The first is on the Radio data set, which obviously has features that the two training methods could not discover. We intend to explore this data set more thoroughly to discover why the two models perform so much worse than the Deep Learning models. The other exception is Reservoir with the WBC data set. The poor performance could be attributed to one of two things: a good reservoir for this problem was not discovered, or the state vector representation (count of neuron fives) used in this paper is not sufficient. We plan to explore different state vector schemas in the future.

In Table 4, we display the neuron and synapse counts for the networks that produced the best classification accuracies. For LSTM, we define a “neuron” as an LSTM cell, which is more complex than a single neuron, but occupies the neuron’s place in the neural network. For all of the Deep Learning models, we define a “synapse” as the number of trainable parameters. For all but the Convolutional model, this is in fact the number of synapses. For the Convolutional model, the actual synapse count is higher, because the convolutional and max-pooling layers train groups of synapses with a single parameter.

6 CONCLUSION

We compare six approaches for training neuromorphic networks for six classification tasks. The six approaches comprised of four Deep Learning approaches, a spiking neural network training approach trained using evolutionary optimization (EONS), and a reservoir computing (RC) approach using spiking neural networks. We found that there was no one clear winner amongst the six approaches on all six tasks in terms of classification accuracy. However, for neuromorphic implementation, classification accuracy is often not the only metric considered; in particular, for certain applications, metrics such as size, weight, and power (SWaP) of the resulting neuromorphic implementation are also important to consider. As a result, we also explore the trade-offs between accuracy and size in Table 4. The results show that the Convolutional model has the highest accuracy, but also the highest size, while the Reservoir model has the lowest size, but also the lowest accuracy. Future work will involve exploring different state vector schemas and parameter settings to improve the performance of the Reservoir model.

Figure 5: Bar graph of model accuracy per data set. Accuracies are from table 3.

Figure 6: Bar graph of network sizes for comparison. Network sizes are in table 4.

With the exception of the Perceptron, which does not classify as well as the others, the Deep Learning networks are orders of magnitude larger than the spiking, recurrent neural networks. Unsurprisingly, the Convolutional model has the most elements, peaking at roughly 20,000 neurons and 3,500,000 synapses for TIMIT. The spiking, recurrent networks are much smaller, highlighting their computational power. With EEG and TIMIT, we explored smaller reservoirs than the standard 200-neuron reservoir, and were able to generate reservoirs that were significantly smaller, yet retained the same classification accuracy. These are included in parentheses in Table 4. It is worth noting that an individual synapse in the spiking recurrent networks is more complex than the individual synapses in the Deep Learning networks. However, since we are targeting building networks for spiking neuromorphic implementations, the synapses in the Deep Learning networks will still take up the same amount of “space” on the chip, even though they are not using the full functionality of the synapses that the spiking recurrent neural networks are.

Table 4. It is worth noting that an individual synapse in the spiking recurrent networks is more complex than the individual synapses in the Deep Learning networks. However, since we are targeting building networks for spiking neuromorphic implementations, the synapses in the Deep Learning networks will still take up the same amount of “space” on the chip, even though they are not using the full functionality of the synapses that the spiking recurrent neural networks are.

6 CONCLUSION

We compare six approaches for training neuromorphic networks for six classification tasks. The six approaches comprised of four Deep Learning approaches, a spiking neural network training approach trained using evolutionary optimization (EONS), and a reservoir computing (RC) approach using spiking neural networks. We found that there was no one clear winner amongst the six approaches on all six tasks in terms of classification accuracy. However, for neuromorphic implementation, classification accuracy is often not the only metric considered; in particular, for certain applications, metrics such as size, weight, and power (SWaP) of the resulting neuromorphic implementation are also important to consider. As a result, we also explore the trade-offs between accuracy and size in Table 4. The results show that the Convolutional model has the highest accuracy, but also the highest size, while the Reservoir model has the lowest size, but also the lowest accuracy. Future work will involve exploring different state vector schemas and parameter settings to improve the performance of the Reservoir model.

Figure 5: Bar graph of model accuracy per data set. Accuracies are from table 3.

Figure 6: Bar graph of network sizes for comparison. Network sizes are in table 4.

With the exception of the Perceptron, which does not classify as well as the others, the Deep Learning networks are orders of magnitude larger than the spiking, recurrent neural networks. Unsurprisingly, the Convolutional model has the most elements, peaking at roughly 20,000 neurons and 3,500,000 synapses for TIMIT. The spiking, recurrent networks are much smaller, highlighting their computational power. With EEG and TIMIT, we explored smaller reservoirs than the standard 200-neuron reservoir, and were able to generate reservoirs that were significantly smaller, yet retained the same classification accuracy. These are included in parentheses in Table 4. It is worth noting that an individual synapse in the spiking recurrent networks is more complex than the individual synapses in the Deep Learning networks. However, since we are targeting building networks for spiking neuromorphic implementations, the synapses in the Deep Learning networks will still take up the same amount of “space” on the chip, even though they are not using the full functionality of the synapses that the spiking recurrent neural networks are.

6 CONCLUSION

We compare six approaches for training neuromorphic networks for six classification tasks. The six approaches comprised of four Deep Learning approaches, a spiking neural network training approach trained using evolutionary optimization (EONS), and a reservoir computing (RC) approach using spiking neural networks. We found that there was no one clear winner amongst the six approaches on all six tasks in terms of classification accuracy. However, for neuromorphic implementation, classification accuracy is often not the only metric considered; in particular, for certain applications, metrics such as size, weight, and power (SWaP) of the resulting neuromorphic implementation are also important to consider. As a result, we also explore the trade-offs between accuracy and size in Table 4. The results show that the Convolutional model has the highest accuracy, but also the highest size, while the Reservoir model has the lowest size, but also the lowest accuracy. Future work will involve exploring different state vector schemas and parameter settings to improve the performance of the Reservoir model.
such, we also compared the sizes of the resulting networks for each of the six approaches. We found that, in general, the spiking approaches (EONS and RC) both produced networks that were orders of magnitude smaller than the networks produced by the successful Deep Learning approaches. Thus, networks trained using those approaches may be more suitable for certain SWaP constrained applications.

It is worth noting that in this work, we approached the comparison of the different methods naively in order to simulate how a non-machine learning expert would approach. There is much additional work that could be done to improve the performance of any given machine learning approach on any given application. Further, this work does not address significantly large datasets (hundreds of thousands of millions or examples) or datasets with many classes. The length of time training with the spiking neural network models (EONS, RC) needs to be compared to the amount of time to train non-spiking variants. We also suspect a comparison of training times between EONS and RC would yield interesting results. Other future work includes exploring extra models, such as Echo State Networks [15]. Because of the promising results for the spiking neural network approaches in terms of resulting network size, we intend to focus our research on those areas moving forward.

ACKNOWLEDGEMENTS

Research sponsored in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy. This research was also supported in part by an Air Force Research Laboratory Information Directorate grant (FA8750-16-1-0065).

REFERENCES


