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

ACM Reference Format:
John J. M. Reynolds, James S. Plank, Catherine D. Schuman, and Grant R. Bruer, Adam W. Disney, Mark E. Dean, Garrett S. Rose. 2018. A Comparison of Neuromorphic Classification Tasks. In *ICONS '18: International Conference on Neuromorphic Systems, July 23–26, 2018, Knoxville, TN, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3229884.3229896

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

Notice: This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with 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 a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan). This research was supported in part by an Air Force Research Laboraty Information Directorate grant (FA8750-16-1-0065).

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. They 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 Eudlidean 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

Name	Software	Neural Network	Training
Perceptron	TENN-Lab	Feed Forward	Back-propagation
MLP	Keras / TensorFlow	Feed Forward	Back-propagation
Conv	Keras / TensorFlow	Feed Forward	Back-propagation
LSTM	Keras / TensorFlow	Nodes with recurrency	Back-propagation
EONS	TENN-Lab / NIDA	Spiking, Recurrent	Evolutionary Optimization
Reservoir	TENN-Lab / NIDA	Spiking, Recurrent + Perceptron	Random + Back-Propagation

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

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.

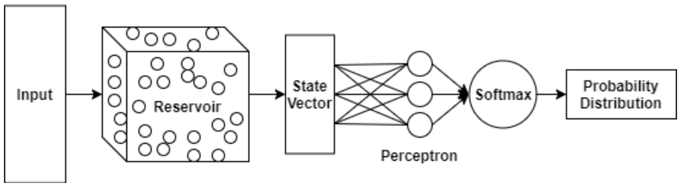

Figure 1: **This is a simple diagram of the reservoir computing paradigm. In our work, the reservoir is made up of a recurrent spiking neural network model known as NIDA. The single layer connected to the weight matrix is the output layer.**

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 feeback 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 Tensor Flow 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.

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.

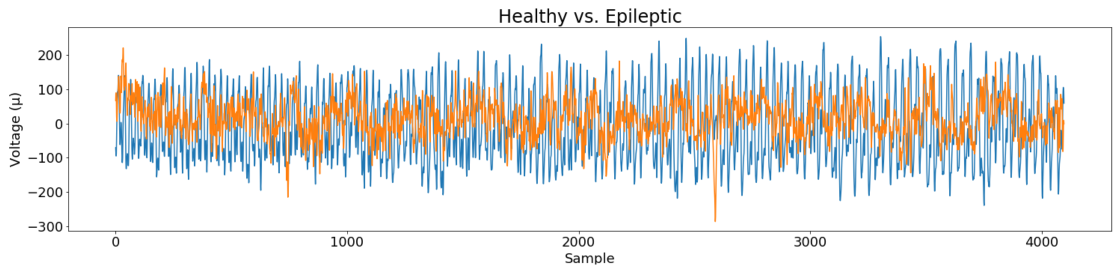

Figure 2: **Example time series plots of two EEG signals: healthy (orange), and epileptic(blue).**

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 is 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.

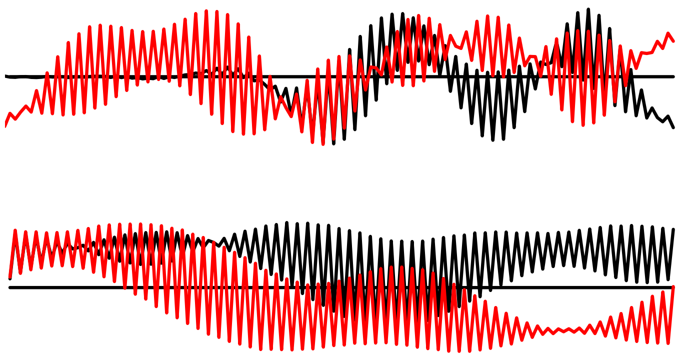

Figure 3: **Two example snippets from the Radio dataset: 8PSK on the top, and GFSK on the bottom. Real values are plotted in black, and imaginary values in red.**

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

Name	r	c	Classifications	Training Data Points	Testing Data Points	N (Time Steps)	M (Output Window)
Iris	4	1	3	75	75	440	330
WBC	9	1	2	560	139	440	330
Pima	8	1	2	615	153	440	330
Radio	2	128	2	667	166	1400	1200
EEG	1	4097	2	160	40	4100	4000
TIMIT	13	48	2	2000	200	500	450

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.

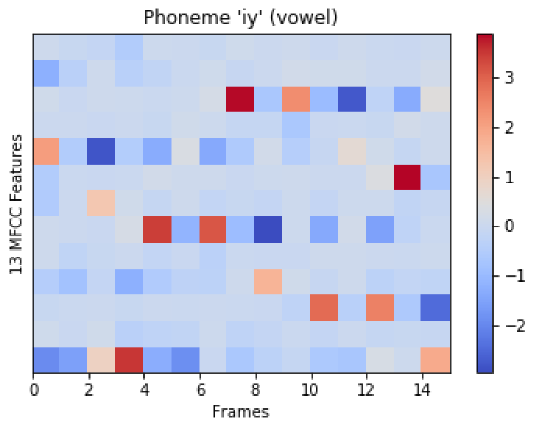

Figure 4: **Spectrogram of MFCC features from a spoken phoneme 'iy'; a vowel. Each column is a MFCC vector with 13 values, calculated every 10 ms over 25 ms of samples.**

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. The best classification accuracy for each data set is on par with, or exceeds previously published data [20, 21, 27, 31].

Model / Data	**Iris**	**WBC**	**Pima**	**Radio**	**EEG**	**TIMIT**
Perceptron	97.38	48.40	67.54	50.70	54.05	77.57
MLP	95.33	94.71	68.37	80.53	48.25	83.73
Conv	96.13	96.45	67.70	93.95	99.00	85.20
LSTM	96.67	95.71	79.22	83.64	45.00	83.40
EONS	98.66	99.28	78.57	71.00	99.00	83.00
Reservoir	93.33	96.47	79.22	73.00	98.00	85.00

Table 3: **Percentage accuracies of each model on each data set. Each number presents the best classification accuracy on the testing set of data.**

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 sigificantly

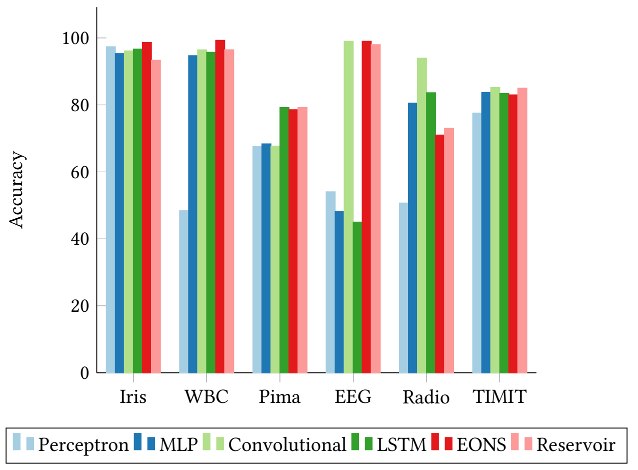

Figure 5: **Bar graph of model accuracy per data set. Accuracies are from table 3.**

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 one exception. The exception 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.

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.

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

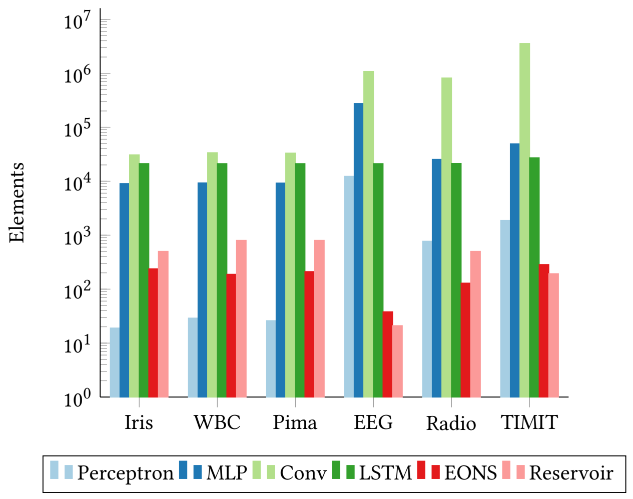

Figure 6: **Bar graph of network sizes for comparison. Network sizes are in table 4**

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 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

Model / Data	Iris		WBC		Pima		Radio		EEG		TIMIT	
	N	S	N	S	N	S	N	S	N	S	N	S
Perceptron	7	12	11	18	10	16	258	512	4099	8194	626	1248
MLP	199	8835	203	9090	202	9026	450	24898	4291	270722	818	48450
Convolutional	775	30019	2678	31042	2314	30786	12278	805652	876547	202562	28006	3530652
LSTM	99	21091	98	21058	98	21058	98	21186	98	21058	98	27074
EONS	74	164	22	166	38	173	26	103	8	30	39	246
Reservoir	200	300	200	600	200	600	200	300	200 (7)	900 (14)	200 (53)	1000 (140)

Table 4: **Neuron and synapse counts for the networks that produced the classification accuracies in Table 3.**

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 or millions of 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 Laboraty Information Directorate grant (FA8750-16-1-0065).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

[2] Luís A Alexandre, Mark J Embrechts, and Jonathan Linton. 2009. Benchmarking reservoir computing on time-independent classification tasks. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE, 89–93.

[3] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. 2001. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E* 64, 6 (2001), 061907.

[4] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer. 2011. Information processing using a single dynamical node as complex system. *Nature Communications* 2 (September 2011).

[5] François Chollet et al. 2015. Keras. https://keras.io. (2015).

[6] Ronald A Cole, Yonghong Yan, Brian Mak, Mark Fanty, and Troy Bailey. 1996. The contribution of consonants versus vowels to word recognition in fluent speech. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, Vol. 2. IEEE, 853–856.

[7] S. B. Davis and P. Mermelstein. 1980. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28, 4 (1980), 357–366.

[8] DeepSig, Inc. 2017. Resources for Radio Machine Learning. https://www.deepsig.io/resources. (2017).

[9] Aida A Ferreira and Teresa B Ludermir. 2009. Genetic algorithm for reservoir computing optimization. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE, 811–815.

[10] R. A. Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annual Eugenics* 7-II (1936), 179–188.

[11] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org

[12] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. *CoRR* abs/1303.5778 (2013). arXiv:1303.5778 http://arxiv.org/abs/1303.5778

[13] T. Hastie, R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY.

[14] S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (December 1997), 1735–1780.

[15] H. Jaeger. 2001. The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology. Reprint in 2010 with erratum note at http://minds.jacobs-university.de/sites/default/files/uploads/papers/EchoStatesTechRep.pdf. (2001).

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (November 1998), 2278 – 2324.

[17] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml

[18] W. Maass, T. Natschläger, and H. Markram. 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation* 14, 11 (2002), 2531–2560.

[19] J. P. Mitchell, G. Bruer, M. E. Dean, J. S. Plank, G. S. Rose, and C. D. Schuman. 2017. NeoN: Neuromorphic Control for Autonomous Robotic Navigation. In *IEEE 5th International Symposium on Robotics and Intelligent Sensors*. Ottawa, Canada, 136–142.

[20] D. Norton and D. Ventura. 2010. Improving liquid state machines through iterative refinement of the reservoir. *Neurocomputing* 73 (2010), 2893–2904.

[21] T. J. O'Shea, J. Corgan, and T. C. Clancy. 2016. Convolutional Radio Modulation Recognition Networks. arXiv:1602.04105v3. (June 2016).

[22] J. S. Plank, G. S. Rose, M. E. Dean, C. D. Schuman, and N. C. Cady. 2017. A Unified Hardware/Software Co-Design Framework for Neuromorphic Computing Devices and Applications. In *IEEE International Conference on Rebooting Computing (ICRC 2017)*. Washington, DC.

[23] C. D. Schuman. 2017. Poster: The Effect of Biologically-Inspired Mechanisms in Spiking Neural Networks for Neuromorphic Implementation. 5th Neuro Inspired Computational Elements Workshop (NICE 2017). (March 2017). https://www.src.org/calendar/e006125/

[24] C. D. Schuman, J. D. Birdwell, and M. E. Dean. 2014. Neuroscience-Inspired Dynamic Architectures. In *Biomedical Science and Engineering Center Conference (BSEC), Oak Ridge National Laboratory*.

[25] C. D. Schuman, J. D. Birdwell, and M. E. Dean. 2014. Spatiotemporal Classification Using Neuroscience-Inspired Dynamic Architectures. *Procedia Computer Science* 41 (2014), 89–97.

[26] C. D. Schuman, A. Disney, S. P. Singh, G. Bruer, J. P. Mitchell, A. Klibisz, and J. S. Plank. 2016. Parallel Evolutionary Optimization for Neuromorphic Network Training. In *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*. IEEE Press, 36–46.

[27] C. D. Schuman, J. S. Plank, A. Disney, and J. Reynolds. 2016. An Evolutionary Optimization Framework for Neural Networks and Neuromorphic Architectures. In *International Joint Conference on Neural Networks*. Vancouver.

[28] Frank Seide and Amit Agarwal. 2016. CNTK: Microsoft's Open-Source Deep-Learning Toolkit. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 2135–2135. https://doi.org/10.1145/2939672.2945397

[29] Nicholas Soures, Lydia Hays, and Dhireesha Kudithipudi. 2017. Robustness of a memristor based liquid state machine. In *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2414–2420.

[30] W. N. Street, W. H. Wolberg, and O.L. Mangasarian. 1993. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology*, Vol. 1 905. San Jose, CA, 861–870.

[31] Abdulhamit Subasi and M Ismail Gursoy. 2010. EEG signal classification using PCA, ICA, LDA and support vector machines. *Expert Systems with Applications* 37, 12 (2010), 8659–8666.

[32] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016). http://arxiv.org/abs/1605.02688
[33] Fang Zheng, Guoliang Zhang, and Zhanjiang Song. 2001. Comparison of different implementations of MFCC. *Journal of Computer Science and Technology* 16, 6 (2001), 582–589.