Deep Spatiotemporal Feature Learning with Application to Image Classification

Thomas P. Karnowski
Image Science and Machine Vision Group
Oak Ridge National Laboratory
Oak Ridge, TN, 37831
Email: karnowskitp@ornl.gov

Itamar Arel, Derek Rose
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN, 37996
Email: itamar@ieee.org, derek@utk.edu

***Abstract*—Deep machine learning is an emerging framework for dealing with complex high-dimensionality data in a hierarchical fashion which draws some inspiration from biological sources. Despite the notable progress made in the field, there remains a need for an architecture that can represent temporal information with the same ease that spatial information is discovered. In this work, we present new results using a recently introduced deep learning architecture called Deep Spatio-Temporal Inference Network (DeSTIN). DeSTIN is a discriminative deep learning architecture that combines concepts from unsupervised learning for dynamic pattern representation together with Bayesian inference. In DeSTIN the spatiotemporal dependencies that exist within the observations are modeled inherently in an unguided manner. Each node models the inputs by means of clustering and simple dynamics modeling while it constructs a belief state over the distribution of sequences using Bayesian inference. We demonstrate that information from the different layers of this hierarchical system can be extracted and utilized for the purpose of pattern classification. Earlier simulation results indicated that the framework is highly promising, consequently in this work we expand DeSTIN to a popular problem, the MNIST data set of handwritten digits. The system as a preprocessor to a neural network achieves a recognition accuracy of 97.98% on this data set. We further show related experimental results pertaining to automatic cluster adaptation and termination.**

***Index Terms*—deep learning; biologically-inspired computing; online clustering;**

I. Introduction

Deep machine learning (DML) is an emerging framework for dealing with complex data in a hierarchical fashion which draws some inspiration from biological sources. Bengio, in [1], provides a comprehensive overview of deep learning theory, where a deep architecture is defined as "composed of multiple levels of non-linear operations, such as in neural nets with many hidden layers or in complicated propositional formulae re-using many sub-formulae." The use of multiple levels of operations can greatly simplify the computational load of a learning architecture, provided it can be successfully trained and optimized for the problem at hand. Two fairly recent deep learning architectures of note are Convolutional Neural Networks (CNNs) [2] and Deep Belief Networks (DBNs) [3]. Convolutional neural networks are discriminative connectionist models designed to operate directly on observed images without preprocessing. They have been proven robust against noise and (reasonable levels of) geometric distortion or transformation in the context of image classification. DBNs are probabilistic generative models that are composed of multiple layers of stochastic, latent variables; traditionally DBNs lack the ability to combine unsupervised learning with supervised learning in a manner that allows unlabeled observations to be learned and represented independently of labeled ones. Recent work by [4] [5] has made great strides in scaling both unsupervised and semi-supervised learning in DBNs, though training of these models remains computationally costly.

In addition, recent neuroscience findings suggest that the neocortex itself has a hierarchical nature of identical building blocks or cortical circuits [6]. Such structure facilitates effective learning and interpretation of sensory information, particularly in the context of capturing spatiotemporal dependencies. As in deep learning, the core assumption is that by partitioning high-dimensional sensory signals into smaller segments and modeling those based on regularities in the observations, a scalable system emerges which is capable of dealing with the virtually infinite (though structurally bound) amount of information mammals are exposed to over time. Physiologically supported by research in the visual area of the cortex [7], work in [8] has introduced a distinct generative Bayesian inference model. Other biologically inspired work includes [9], [10], [11] among others.

Thus the concept of partitioning large data structures into smaller, more manageable units, and discovering the dependencies that may or may not exist between such units, is very promising. However, there remains a need for an architecture that can represent temporal information with the same ease in which spatial structure is discovered. Moreover, some key constraints are imposed on the learning schemes driving these architectures, namely the need for layer-by-layer training, and often times pre-training. In this work, we present results from a novel deep learning architecture, the Deep Spatio-Temporal Inference Network (DeSTIN). As presented in earlier work [12] and [13], DeSTIN is a novel discriminative deep learning architecture that combines concepts from unsupervised learning for dynamic pattern representation together with Bayesian inference. In DeSTIN the spatiotemporal dependencies that exist within the observations are modeled inherently in an unguided manner. Each node in the hierarchy models the inputs by means of clustering and simple dynamics modeling, while

it constructs a belief state over the distribution of sequences using Bayesian inference. We demonstrate that information from the different layers of this hierarchical system can be extracted and utilized for the purpose of pattern classification. In this work we expand DeSTIN to a popular problem, the MNIST data set of handwritten digits [14], which is widely used for various machine learning algorithms.

In the following sections we review core concepts pertaining to the DeSTIN architecture. We discuss the main learning mechanisms along with key metrics and parameters for formulating those mechanisms. We then discuss the experimental configuration for the MNIST data set and show results from using DeSTIN as a feature extraction engine for the problem set. Finally, we conclude with discussion and summary of projected future directions for our work.

II. Technical Approach

We summarize the key elements of DeSTIN here and refer to [12] and [13] for more detail. The architecture contains a hierarchy of layers whereby each layer consists of multiple instantiations of an identical circuit or node. Each node observes and learns to represent a temporal sequence of patterns. The lowest layer of the hierarchy processes temporally changing input data, such as image pixels, and over time continuously constructs a belief state that attempts to characterize the sequences of patterns viewed. The second layer, and all those above it, receive as input the belief states of nodes at their corresponding lower layers, and attempt to construct belief states that capture regularities in their inputs. The architecture thus forms as outputs at each node hierarchical belief states across all its layers which captures both spatial and temporal regularities in the data - a novel, key advantage over existing deep learning schemes. These outputs can be fed to a supervised learning algorithm (such as a neural network) to perform classification. In addition, since each node is identical, the architecture can be mapped to parallel computational platforms such as graphics processing units. The overall processing is simple and does not rely on large amounts of memory which makes it tractable for hardware-oriented approaches as well. Each node in DeSTIN maps its current belief and observation to a new belief state that thus reflects a longer temporal pattern or sequence. Finally, feedback from the upper-layer (or parent) node is received and utilized in the formulation of the belief state.

The fundamental belief update rule of DeSTIN was derived in [12] and is given as

$$b'(s') = \frac{\Pr(o|s') \sum_{s' \in S} \Pr(s'|s,a) b(s)}{\sum_{s'' \in S} \Pr(o|s'') \sum_{s' \in S} \Pr(s''|s,a) b(s)} \quad (1)$$

which maps the current observation o, the belief b (with argument the system state s) and the belief state or advice of a higher-layer node a, to a new (updated) belief and state $b\prime(s\prime)$ at the next time step. The denominator term is essentially a normalization factor. One interpretation of this equation is that the (static) pattern similarity metric, $\Pr(o|s')$, is modulated by a construct that reflects the system dynamics, $\sum_{s' \in S} \Pr(s'|s,a) b(s)$. (For shorthand, the latter is denoted as PSSA.) As such, the belief state inherently captures both spatial and temporal information and these two constructs are the main items which must be learned from the data. The former is learned using online clustering, while the latter is learned from experience by adjusting of the parameters with each transition from s to $s\prime$ given a. In past implementations, the advice or belief of the parent node, a, was chosen using the selection rule of $a = \arg\max_s b_p(s)$. The result is a robust framework that autonomously learns to represent complex data patterns, such as those found in real-life robotics applications and whose output can be used as a generic feature extractor for a supervised learning system.

The online clustering algorithm is the core of the learning process for each node and includes constructs for improving performance and modulating the learning rate, as discussed in [15]. The basic clustering algorithm uses the winner-take-all (WTA) competitive learning approach, however the centroids are continuously updated online based on the input observations. Since the goal is to produce a system which can scale efficiently with simple hardware, it is assumed the system cannot iterate with the entire data set in memory while converging to the cluster centroids. Also, a finite, fixed number of centroids are assumed. In competitive learning clustering algorithms the learning rate is often adjusted to allow trade offs between faster learning in early phases of iterations and stability in later phases. Typically the learning rate is adjusted so that it is monotonically decreasing, for example a decaying exponential with the decay as a function of iteration. In past work we have experimented with adaptive learning rates which worked well for simple problems. Another option is to choose a constant learning rate. Regardless, the update rule for the winning centroid x is achieved by

$$x^{t+1} = x - \alpha \|x - o\| \quad (2)$$

where α represents the learning rate and o is the observation or input vector. Also incorporated is a "starvation trace" mechanism, which is used to include clusters which are initially too far from observations to be updated. The starvation trace allows idle or starved centroids to accumulate credit over time when they are not the selected centroid (and lose credit when they are the selected centroid). The starvation trace is initialized to a constant vector of length D where D is the dimensionality of the observation. For each observation where centroid x is chosen, the non-selected centroids "accumulate credit" by having their starvation trace value decreased by a small constant. The starvation trace is employed to weight the distance calculation and thus render "starved" clusters a chance to make movement towards data samples. A summary description of the algorithm is as follows. The estimated centroids are initialized to random values. A new observation is then assigned to a single estimated centroid based on the

minimum distance computed by some similarity value such as Euclidean distance. This distance metric is weighted by the starvation trace as:

$$d_x = dist(x, o) = \|x - o\|\{1 - starve_x\} \tag{3}$$

where $starve_x$ is the starvation trace. Thus as the starvation trace increases, the distance metric appears to decrease and gives the "starved centroid" an opportunity for updates. The WTA centroid selection simply performs $\arg\min_x d_x$ to select the winning centroid for updates, however the algorithm always chooses the labeling centroid as the one with closest distance (without regard to starvation trace).

A set of metrics of interest are the mean and standard deviation of the changes made in the centroids over time which can indicate that the centroids have reached a relatively stable point, as the mean should reach some constant or near-constant value while the standard deviation gives a measure of the spread of the vectors that are drawn to that centroid. A function of these two values generates a single value which can be used to either adjust the learning rate or terminate clustering. These values are all computed on-line and are given by the following formulas where d_x is the distance between an observation and winning centroid x:

$$\mu_x^{t+1} = a\mu_x^t + (1 - a)d_x \tag{4}$$

$$\sigma_x^{t+1} = b\sigma_x^t + (1 - b)\|\mu_x^t - d_x\| \tag{5}$$

where a and b are constants less than 1. Both vectors are initialized by computing the running mean and standard deviation approximation on the initial N updates, where $N = \frac{1}{a}$ or $N = \frac{1}{b}$ as appropriate. Ideally as the winning centroid comes to represent the actual centroid the value of the mean change estimate μ_x should approach 0 (since d_x should become smaller and smaller) and the value of the change standard deviation estimate σ_x should also approach 0 (because both d_x and μ should become smaller and smaller). The combination of the metrics is expressed as

$$\hat{\rho}(\mu_x, \sigma_x) = \frac{2}{1 + e^{-\gamma \frac{\sigma_x}{1+\mu_x}}} - 1 \tag{6}$$

and this in turn is also windowed as is μ_x and σ_x but with an initial value of unity. These functions will produce small values when the centroids are not changing or are changing relatively small amounts, while periods of large change in centroids will force $\hat{\rho}$ to be nearly unity.

As a final note, the belief estimate is computed from the clustering outputs using the equation

$$b(s) = 1 - \frac{d_s}{\sum_{s'' \in S} d_{s''}} \tag{7}$$

This expresion takes the distance of centroid s to the input vector and normalizes by the sum of the distances to all centroids so when d_s is small (i.e., 0) there is high "belief" that this centroid is the correct one.

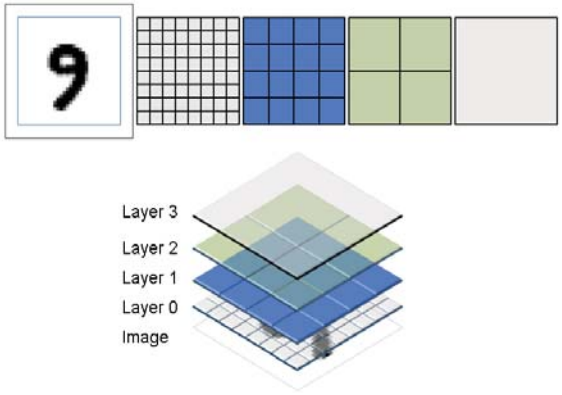

Fig. 1. DeSTIN Hierarchy for the MNIST dataset studies. Four layers are used with 64, 16, 4 and 1 node per layer arranged in a hierarchical manner. At each node the output belief b(s) at each temporal step is fed to a parent node. At each temporal step the parent receives input beliefs from four child nodes to generate its own belief (fed to its parent) and an advice value a which is fed back to the child nodes.

III. Experiments and Results

Initial verification of the DeSTIN approach [12],[13] used a very simple study using three different alphabetic characters. In this work we present the results of DeSTIN analysis of the MNIST database of handwritten characters [14]. The dataset consists of 60,000 training images and 10,000 test images. The best performance of reported machine learning methods [16] achieve over 99% accuracy, and even simple machine learning algorithms such as kNN classifier perform quite well on the dataset (95% accuracy [2]). A fusion of several different methods and comparison with humans reveals the best possible performance is likely 99.8% [17] . For DeSTIN, the MNIST dataset images were padded from 28 x 28 pixels in size to 32 x 32 pixels. A hierarchy of 4 layers of sizes 8 x 8 nodes, 4 x 4 nodes, 2 x 2 nodes, and a single node at the top layer was used as depicted in Figure 1. At the lowest layer each input node is presented a 4 x 4 pixel region of the input image. Each layer uses a different number of centroids, choosing 25, 16, 12, and 10 for each layer. The image is then shifted through a sequence of 64 different movements which are offset by a single pixel and form a serpentine pattern and emulate sacchading of the human vision system. The movement ranged from (0,0) pixels to (7,7) total pixels, so the input image was padded to cover boundary regions. The movement pattern was not optimized and may not be the best sequence for a complete online system that iteratively derives a best belief for the input image but served as a good case for initial study. In addition the "PSSA" computation from equation 1 was omitted; thus the extracted features are from the clustering performed at different layers in the hierarchy on the beliefs computed by equation 7.

A. Clustering Metrics

The first experiments used a sampling of the MNIST training set (every 25th image) and examined the effect of

TABLE I
NUMBER OF UNIQUE VECTORS IN LAYER0 PER ROW,COLUMN FOR CLUSTER METRIC STUDY

	0	1	2	3	4	5	6	7
0	1	46	390	1351	2218	1567	431	29
1	120	2145	11680	29383	41219	32262	12306	1687
2	636	9417	40725	81295	98904	79600	34484	5412
3	982	15821	59799	99964	111033	90396	39300	5818
4	925	18497	63472	99299	109267	86724	35705	5068
5	1205	19782	63053	98565	106894	77893	29735	4255
6	857	13428	47881	82299	83252	49171	14828	1794
7	159	3297	15408	29495	27334	12642	2733	275

TABLE II
CLUSTER STOPPING POINTS FOR NODES OF LAYER 0

	0	1	2	3	4	5	6	7
0	N	N	N	N	N	N	N	N
1	N	N	N	N	N	N	N	N
2	N	N	N	3002	2511	2985	N	N
3	N	N	3883	2598	2423	2739	N	N
4	N	N	4808	2715	2383	2980	N	N
5	N	N	3554	2616	2418	3393	N	N
6	N	N	4663	2949	3203	N	N	N
7	N	N	N	N	N	N	N	N

TABLE III
CLUSTER STOPPING POINTS FOR NODES OF LAYER 1

	0	1	2	3
0	N	N	9337	N
1	N	6712	6647	8400
2	N	6751	6642	N
3	N	7092	6898	N

learning rate on the values of μ and ρ for layers 0 and 3. For layer 0, node (3,4) is reviewed as this node sees the most variation from the input sequence (see Table I). The top layer is chosen also as it gives the highest "overview" of the entire processed sequence. These are plotted in Figures 2 and 3 with the observation number on the x-axis (where there are 64 movements or observations per input MNIST digit) for learning rates of 0.001 and 0.0001. These plots show that little is gained in the sense of the clustering stability after roughly 6000 observations for the first layer. However the final layer shows that the mean change increases for some centroids, indicating that they are not in a stable position but the change relative to the standard deviation indicates the centroid learning may be reaching a reasonable bounding value. For the slower learning rate, the value of ρ shows that the learning takes longer, as expected, since the value of the largest entry does not reach a comparable level to the faster rate until around 9000 observations. The value of μ at the top layer shows more erratic behavior, reaching a plateau around 6000 then increasing throughout the rest of the sequences before decreasing again after about 10000 observations. This is likely due to the stabilization of most of the lower level nodes around 6000 which causes the highest layer to settle

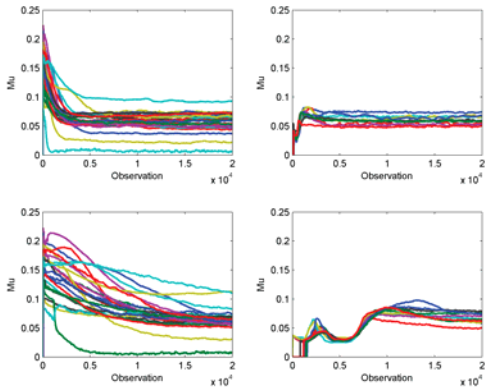

Fig. 2. Top: Mu for learning rate of 0.001. Left is bottom layer, right is top layer. Bottom: Mu for learning rate of 0.0001. Left is bottom layer, right is top layer.

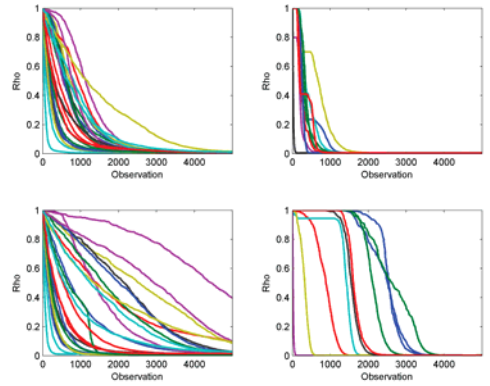

Fig. 3. Top: Rho for learning rate of 0.001. Left is bottom layer, right is top layer. Bottom: Rho for learning rate of 0.0001. Left is bottom layer, right is top layer.

a bit, but later observations cause additional changes that are not as well matched and thus the centroids drift again.

An online error was generated by computing the difference between each input vector and the adjusted, winning centroid. These plots are shown in Figure 4 for layers 0 and 3 again. A smoothing window is applied to the plots of size 64. For layer

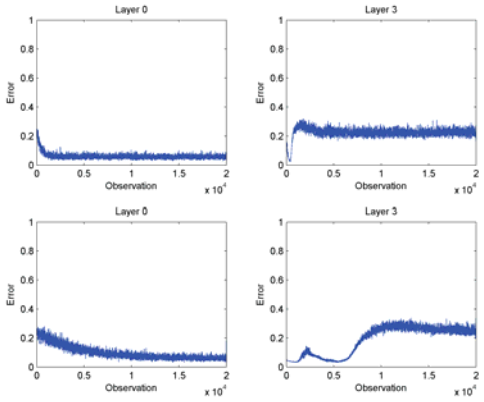

Fig. 4. Top: error with learning rate at 0.001, with (left) layer 0 and (right) layer 3. Bottom: error with learning rate at 0.0001, with (left) layer 0 and (right) layer 3.

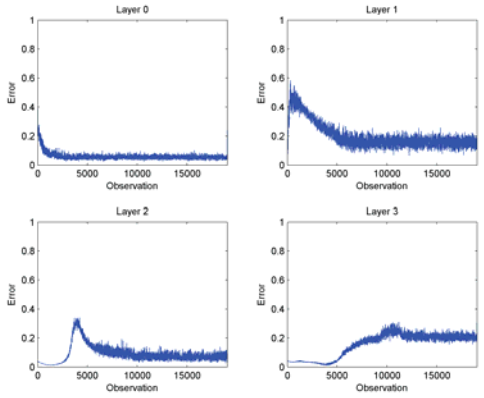

Fig. 5. Error rates for node (3,4) of layer 0, node (1,2) of layer 1, (0,1) of layer 2, and (0,0) of layer 3 using the adaptive learning rate and automatic termination of clustering.

0, we see that the error decreases fairly rapidly to a roughly constant level. For the top layer, we see that a sort of minimum error is reached rather early in the sequence but the smoothed error increases to a significantly larger amount than layer 0. The minimum error is likely where the online clustering has reached a good match relative to the immature response of the lower layers. The higher upper level error can be partially explained from the smaller number of centroids (10 instead of 25) at this layer. With the lower learning rate, we see a similar behavior for layer 0 although the error decline is slower. The top layer in the slower learning rate case shows an increase in the error after reaching a sort of plateau as seen in plots of μ as well. The error then begins to decline gradually.

In a final experiment the learning rate was adaptively modified the learning rate and criteria studied to automatically terminate clustering. In this schema, we evaluated the innermost nodes of the initial layer and all nodes of subsequent layers. (As shown in Table I, the edge nodes of layer 0 do not show as much variation.) The learning rate was initialized to 0.001 for layer 0 and 0.0001 for subsequent layers. The mean across all centroids for ρ was computed at each observation. When the mean value was less than 0.05, clustering terminates for the node. When half the nodes were terminated the entire layer clustering was stopped. The learning rate for the next layer was then reduced to 0.001 and an additional 1000 non-monitored digit presentations were performed followed by renewed monitoring of the value of ρ. This process was repeated until the top layer clustering was terminated. The clustering termination point is shown in Tables II and III for each node. Note that "N" denotes a node that either was not included or did not finish clustering before half the candidate notes completed. The top layer stopped at 14050 digits and layer 2 stopped at 10984 and 11043 digits (nodes (0,1) and (1,0)). The resulting error plots are shown in Figure 5. The first layer response is as expected, with the error dropping quickly to a fairly constant value. The second layer response is more complex as the error increases initially then slowly drops, settling out around digit presentation 5000. In fact, the clustering of layer 0 stops adapting after digit presentation 4808, so layer 1 adapts slowly with the slow learning rate, then the increase in the learning rate does not seem to change the response much. Layer 1 has its clustering frozen at presentation 8400 and we see that shortly afterward layer 2 reaches a relative constant value. However, the first 4000 or so presentations to layer 2 have a very low error rate. In this case, analysis of the data showed that the reponse was dominated by a single cluster which closely matched the output of layer 1, but was somewhat meaningless because layer 1 had not begin to adapt. Once layer 1 adapts, we see that the error of layer 2 begins to increase to a peak around presentation 4000 and then it declines as layer 1 stabilizes. A similar phenomena is shown for the top layer, although its adaptation is not as pronounced after approximately presentation 12000.

B. Supervised Learning

For the next experiment we used the adjustable clustering stop criteria described in the previous section. After clustering terminated for all layers, the entire training set and testing set was presented to the DeSTIN network and the output belief states at each of the 64 movements was saved. These were temporally sampled with a period of 12 and the training set was used to train a neural network using the MATLAB Neural Network toolbox. The neural network used two hidden layers of 40 nodes each and was trained by using the training set split into a true training set (using 70% of the input images) and a validation set of 30% to prevent overtraining. The data set consisted of the nodes of the top 3 layers and the bottom inner nodes (excluding nodes on the edge of the image). Ten different network training sequences were used and the results were combined in a simple voting scheme, with ties resolved by taking the class with the maximum summed neural network response. The resulting composite performance was 97.98% accuracy (2.02% error) as shown in Table IV. We note that our experimental results are significantly better than a basic kNN

TABLE IV
CONFUSION MATRIX FOR MNIST DATA SET

	0	1	2	3	4	5	6	7	8	9	Perf
0	975	1	0	0	0	0	2	1	1	0	99.49%
1	0	1126	1	1	0	1	2	1	3	0	99.21%
2	5	2	1011	4	1	0	0	8	1	0	97.96%
3	0	0	4	988	0	4	0	4	10	0	97.82%
4	0	0	1	0	963	0	4	1	2	11	98.06%
5	2	0	1	4	0	873	4	2	4	2	97.87%
6	3	2	1	0	1	8	939	0	4	0	98.02%
7	1	3	11	2	0	0	0	1004	2	5	97.66%
8	3	0	4	6	4	4	1	5	941	6	96.61%
9	2	3	1	4	6	2	0	4	9	978	96.93%

classifier applied to the image data, but are not at the state-of-the-art for this data set. Furthermore, running the same neural network configuration and voting scheme on the raw image pixels produced a performance of 96.35% so we are confident DeSTIN is serving as a beneficial feature extractor. Overall, our results are encouraging and represent a significant step in validating our approach over our initial test case. We suspect better performance could be obtained by sampling more of the movements but our overall focus is more on the online learning and autonomous aspect of the architecture so we do not believe this would be useful for our ultimate goals for the architecture and research.

IV. CONCLUSION

We have presented the concept of a deep learning spatio-temporal inferencing architecture which is well-suited for imaging applications. Since earlier work on this architecture focused on a simple test problem, this work has focused on testing our concepts on the more complex MNIST data set. Our results show the system is capable of extracting features suitable for input to a standard neural network architecture and delivers good performance. There are still several open issues. We would like to determine a less heuristic grounds for cluster termination or adaptive learning rate application. In addition, for future work, we seek to expand the inferencing to a more on-line, continuous system which can more readily focus the results in a temporal sense to a final, stable estimate of the hidden state. While we do not foresee the absence of a supervised classifer for the final output, we believe we can make progress towards autonomous learning that can easily be mapped to supervised labels through more computationally simple supervised learning methods.

ACKNOWLEDGMENT

The authors would like to thank Max Mueller, Bobby Coop and Steven Young of the University of Tennessee - Knoxville Machine Intelligence Laboratory and Ryan Kerekes and Ethan Farquhar of the Measurement Science and Systems Engineering Division of Oak Ridge National Laboratory for help in performing the neural network evaluations.

REFERENCES

[1] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[4] H. Lee, R. Grosse, R. Ranganath, and A. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[5] H. Lee, Y. Largman, P. Pham, and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," *Advances in neural information processing systems*, vol. 22, pp. 1096–1104, 2009.

[6] D. Felleman and D. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral cortex*, vol. 1, no. 1, p. 1, 1991.

[7] T. Lee, D. Mumford, R. Romero, and V. Lamme, "The role of the primary visual cortex in higher level vision," *Vision research*, vol. 38, no. 15-16, pp. 2429–2454, 1998.

[8] D. George, "How the brain might work: A hierarchical and temporal model for learning and recognition," Ph.D. dissertation, Stanford University, 2008.

[9] T. Dean, G. Carroll, and R. Washington, "On the prospects for building a working model of the visual cortex," in *Proceedings of the National Conference on Artificial Intellligence*, vol. 22, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, p. 1597.

[10] J. Miller and P. Lommel, "Biomimetic sensory abstraction using hierarchical quilted self-organizing maps," in *Proceedings-SPIE the International Society for Optical Engineering*, vol. 6384, 2006, p. 638.

[11] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 411–426, 2007.

[12] I. Arel, D. Rose, and T. Karnowski, "A Deep Learning Architecture Comprising Homogeneous Cortical Circuits for Scalable Spatiotemporal Pattern Inference," in *NIPS 2009 Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[13] I. Arel, D. Rose, and R. Coop, "DeSTIN: A Scalable Deep Learning Architecture with Application to High-Dimensional Robust Pattern Recognition," in *Proc. of the AAAI 2009 Fall Symposium on Biologically Inspired Cognitive Architectures (BICA)*, 2009.

[14] Y. LeCun and C. Cortes, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 2009.

[15] S. Young, I. Arel, T. Karnowski, and D. Rose, "A Fast and Stable Incremental Clustering Algorithm," in *2010 Seventh International Conference on Information Technology*. IEEE, 2010, pp. 204–209.

[16] K. Labusch, E. Barth, and T. Martinetz, "Simple method for high-performance digit recognition based on sparse coding," *IEEE Transactions on Neural Networks*, vol. 19, no. 11, pp. 1985–1989, 2008.

[17] D. Keysers, "Comparison and Combination of State-of-the-art Techniques for Handwritten Character Recognition: Topping the MNIST Benchmark," *Arxiv preprint arXiv:0710.2231*, 2007.