Image Classification using Generative NeuroEvolution for Deep Learning

Phillip Verbancsics & Josh Harguess
Space and Naval Warfare Systems Center – Pacific
San Diego, CA 92152
{phillip.verbancsics,joshua.harguess}@navy.mil

Abstract

Research into deep learning has demonstrated performance competitive with humans on some visual tasks; however, these systems have been primarily trained through supervised and unsupervised learning algorithms. Alternatively, research is showing that evolution may have a significant role in the development of visual systems. Thus neuro-evolution for deep learning is investigated in this paper. In particular, the Hypercube-based NeuroEvolution of Augmenting Topologies is a NE approach that can effectively learn large neural structures by training an indirect encoding that compresses the artificial neural network (ANN) weight pattern as a function of geometry. The methodologies are tested on a traditional image classification task as well as one tailored to overhead satellite imagery. The results show that HyperNEAT struggles with performing image classification by itself, but can be effective in training a feature extractor that other ML approaches can learn from. Thus NeuroEvolution combined with other ML methods provides an intriguing area of research that can replicate the processes in nature.

1. Introduction

Evolution has a significant role in creating biological visual systems [16, 6], thus a potential path for training artificial vision as effective as the biological counterparts is neuro-evolution (NE; [29]) . However, NE has been challenged because of the required structure sizes. While the human brain includes trillions of connections [14, 30], traditional NE approaches produce networks significantly smaller in size [29, 23]. Thus an active area of research is evolutionary approaches that address this gap [23]. One significant area of interest is generative and developmental systems (GDS), which are inspired by principles that map genotype to phenotype in nature. Such principles are enabled by reusing genetic information [14]. For example, biological neural networks exhibit several important organizational principles, such as modularity, regularity, and hierarchy [9, 27], which can be exploited through the indirect encoding and enable evolution of complex structures.

The GDS approach investigated in this paper is called the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT; [23]). HyperNEAT trains an indirect encoding, called a *compositional pattern producing network* (CPPN; [22]) that represents artificial neural network (ANN) connection weights. HyperNEAT has shown promise in simple visual discrimination tasks [5, 3, 26, 10], but has been limited to two or fewer hidden layers. This paper extends the investigation to deeper architectures by applying HyperNEAT to the MNIST benchmark dataset. Interestingly, the results show that HyperNEAT alone has difficulty in classifying the images, but HyperNEAT is effective at training a feature extractor for backward propagation learning. That is, HyperNEAT can learn the 'deep' parts of the neural network, while backward propagation can perform the fine tuning to make classifications from the generated features. This capability is further demonstrated on the BCCT200 data set.

2. Background

The geometry-based methods that underlie the approach are described in this section.

2.1. NeuroEvolution of Augmenting Topologies (NEAT) and HyperNEAT

The NeuroEvolution of Augmenting Topologies (NEAT) algorithm [24] is a popular neuroevolutionary approach that has been proven in a variety of challenging tasks, including particle physics [1] and function approximation [28], among others [24]. NEAT starts with a population of simple ANNs that increase their complexity over generations by adding structure. The techniques that facilitate evolving a population of diverse and increasingly complex networks are described in detail in Stanley and Miikkulainen [24]; the important concept for the approach in this paper is that NEAT is an evolutionary method that discovers the right topology and weights of a network to maximize performance on a task.

978-1-4799-6683-7/15 $31.00 © 2015 IEEE
DOI 10.1109/WACV.2015.71

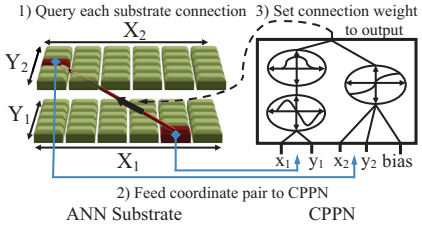


Figure 1. **A CPPN Describes Connectivity.** A grid of nodes, called the ANN *substrate*, is assigned coordinates. (1) Every connection between layers in the substrate is queried by the CPPN to determine its weight; the line connecting layers in the substrate represents a sample such connection. (2) For each such query, the CPPN inputs the coordinates of the two endpoints. (3) The weight between them is output by the CPPN.

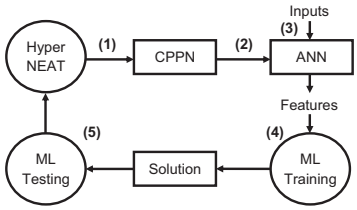


Figure 2. **HyperNEAT Feature Learning.** To learn features, HyperNEAT trains CPPNs (1) that generate the connectivity for a defined ANN substrate (2). The ANN substrate processes the inputs from a data set to produced a set of features (3). These features are given to another machine learning algorithm (4) that learns to perform the task (e.g. image classification). Machine learning then produces a solution that is evaluated on testing data (5). The performance of the solution on data not seen during training provides the fitness score of the CPPN for HyperNEAT.

Hypercube-based NEAT (HyperNEAT; [23]) is a GDS extension of NEAT that enables effective evolution of high-dimensional ANNs. The effectiveness of the geometry-based learning in HyperNEAT has been demonstrated in multiple domains, such as multi-agent predator prey [4] and RoboCup Keepaway [25]. A full description of HyperNEAT is in Stanley et al. [23]. The main idea in HyperNEAT is that geometric relationships are learned through an indirect encoding that describes how the *weights* of the ANN can be *generated* as a function of geometry, as shown in Figure 1. Unlike a direct representation, wherein every connection in the ANN is described individually, an indirect representation describes a pattern of parameters without explicitly enumerating each such parameter. HyperNEAT discovers the *regularities* in the geometry and learns from them. The indirect encoding in HyperNEAT is called a *compositional pattern producing network* (CPPN; [22]), which encodes the *weight pattern* of an ANN [23]. The idea behind CPPNs is that geometric patterns can be encoded by a *composition of functions* that are chosen to represent common regularities.

In summary, HyperNEAT evolves the topology and weights of the CPPN that *encodes* ANN weight patterns. The next section reviews deep learning that has been applied to image classification.

2.2. Deep Learning in Image Classification

Deep learning approaches have achieved remarkable performance in a number of object recognition benchmarks, often achieving the current best performance on these tasks [11, 19, 2, 15]. The challenge for deep learning is how to effectively train such large neural structures. Popular approaches include pre-training the deep architecture through unsupervised learning. Restricted Boltzmann Machines [21] and auto-encoders [12] are among the common approaches to performing this unsupervised training. Alternatively, supervised learning performance can be enhanced through the selection of the deep architecture, such as *convolutional neural networks* (CNNs), which are inspired by proposed models of the human visual cortex [13] and the concept of local receptive fields that take advantage of the input topology. That is, CNNs enforce a particular geometric knowledge by constructing an architecture that learns features based upon locality. The next section introduces approaches to deep learning through HyperNEAT.

3. Approach: Deep Learning HyperNEAT

Although the HyperNEAT method succeeds in a number of challenging tasks [23, 25] by exploiting geometric regularities, it has not yet been applied to tasks where deep learning shows promise or deep architectures. Because HyperNEAT learns as a function of geometry, it is well-suited towards architectures similar to CNNs. This approach introduces two modifications to HyperNEAT training.

The first modification introduces the idea of HyperNEAT as a feature learner. Conventional HyperNEAT trains a CPPN that defines an ANN that is the solution, that is, the produced ANN is applied directly to the task and then the ANN's performance on that task determines the CPPN's fitness score. However, this HyperNEAT modification trains an ANN that transforms inputs into features based upon domain geometry and then the features are given to another machine learning approach to solve the problem, the performance of this learned solution then defines the fitness score of the CPPN for HyperNEAT (Figure 2). In this way, HyperNEAT acts as a reinforcement learning approach that determines the best features to extract for another machine learning approach to maximize performance on the task.

The second modification introduces alternative architectures to HyperNEAT. Traditionally, HyperNEAT produces the weight pattern for a ANN substrate that is feed forward, fully connected, and containing only sigmoid activation functions. However, the only information HyperNEAT sees about the substrate is the geometric coordinates of the

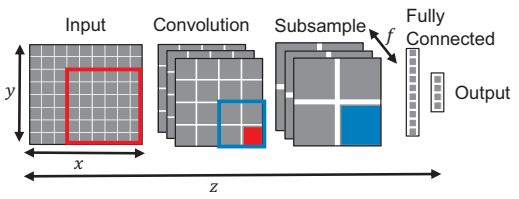


Figure 3. **CNN Substrate for HyperNEAT.** The CNN substrate differs from the traditional HyperNEAT substrate in that it constrains the connectivity from prior layers to a receptive field (in this example a 6×6 followed by 2×2 field). In addition, the CNN alternates a process of convolving features and subsampling features until reduced to a number that can be fully connected.

neurons, thus HyperNEAT could be applied to any graph like structure wherein the nodes have coordinates associated with them. Because CNNs have been demonstrably successful in a number of domains, HyperNEAT is extended to include this alternative ANN architecture as shown in Figure 3. Each of these extensions to HyperNEAT is explored in experiments described in the next section.

4. Experimental Setup

These investigations are conducted on two image classification datasets. The MNIST dataset is first used, which is a popular benchmark for machine learning because it is a challenging and relevant real world problem. MNIST is a set of 28×28 pixel images of handwritten digits $(0-9)$, separated into 60,000 training images and 10,000 testing images. The goal for machine learning is to correctly classify the handwritten digit contained within each image. The methodologies are also tested on the BCCT200 ship recognition dataset, which is a much smaller scale dataset with 4 classes (barge, cargo, container, and tanker) of 200 image each, but is challenging and relevant to a recent real world problem [8, 18]. To this end, HyperNEAT is tested with and without deep learning modifications on the benchmark MNIST. Thus the combinations explored are traditional HyperNEAT training a fully-connected ANN architecture as a classifier, HyperNEAT training an ANN architecture as a feature extractor (ANN that transforms images into features), HyperNEAT training a CNN architecture as a classifier, and HyperNEAT training a CNN architecture as a feature extractor. The most successful of the four (HyperNEAT as a feature extractor with CNN architecture) is applied to the BCCT200 dataset.

The architecture for HyperNEAT in these experiments is a multi-layer neural network wherein the layers travel along the z-axis, each layer consists of a number of features (f-axis), and each feature has a constellation of neurons on the x, y-plane corresponding to pixel locations. Thus the CPPN represents points in an eight-dimensional Hyper-cube that correspond to connections in the four-dimensional substrate and each neuron is located at a particular (x, y, f, z) coordinate. Each layer is presented by a triple (X, Y, F), wherein F is the number of features and X, Y are the pixel dimensions. Thus the input layer is $(28, 28, 1)$, because the input image is 28×28 and contains only grayscale pixel values. The traditional HyperNEAT architecture for these experiments is a seven layer neural network with one input, one output, and five hidden layers, represented by the triples $(28, 28, 1)$, $(16, 16, 3)$, $(8, 8, 3)$, $(6, 6, 8)$, $(3, 3, 8)$, $(1, 1, 100)$, $(1, 1, 64)$, and $(1, 1, 10)$. Each layer in the traditional HyperNEAT ANN architecture is fully connected to the adjacent layers and each neuron has a bipolar sigmoid activation function. The CNN architecture to which HyperNEAT is applied replicates the LeNet-5 architecture that was previously applied to the MNIST dataset [17].

To operate as a feature extractor, the above architectures are modified such that the ANN substrate is cut off before the last hidden layer, that is, for the traditional ANN architecture the $(1, 1, 100)$ layer becomes the new output layer and for the CNN architecture the $(1, 1, 120)$ layer becomes the outputs. Thus each image is passed through the ANN substrate architecture to produce an associated feature vector. For MNIST, these feature vectors are then given to backward propagation to train an ANN with an architecture identical to the architecture that was removed from the substrates. For the BCCT200 data set, these features are classified through the K-Nearest Neighbors (KNN) approach ($k = 30$). The next section presents the results of these HyperNEAT variants.

5. Results

This section will describe the results of the proposed methodology on the MNIST and BCCT200 datasets.

5.1. MNIST

For each of these experiments, results are averaged over 30 independent runs of 2500 generations with a HyperNEAT population size of 256. The fitness score is the sum of the true positive rate, true negative rate, positive predictive value, negative predictive value, and accuracy for each class plus the fraction correctly classified overall and the inverse of the mean square error from the correct label outputs. Each run randomly selects 300 images, evenly spread across the classes, from the MNIST training set for training. For regular HyperNEAT (i.e. not feature learning), fitness is determined by applying the ANN substrate to the training images. For HyperNEAT feature learning an additional 1000 images are randomly selected (again evenly spread across classes) from the MNIST training set. Backward propagation training is run for 250 epochs on the 300 selected images and then tested on the different set of 1000 images. The testing performance of the backward propagation trained ANN becomes the fitness of the CPPN for Hy-

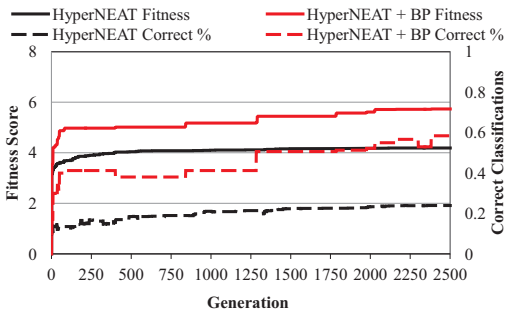


Figure 4. **HyperNEAT Performance with Traditional ANN Architecture.**

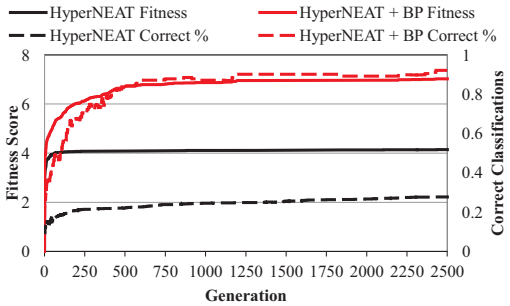
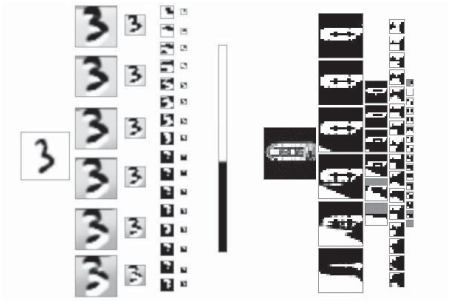


Figure 5. **HyperNEAT Performance with CNN Architecture.**

perNEAT. After evolution completes, the generation champions are evaluated on the MNIST testing set. In the HyperNEAT for feature learning case, the backward propagation trained ANN learns from the full MNIST training set.

As seen in Figure 4, HyperNEAT by itself quickly plateaus at a particular performance level and only gradually learns, achieving an average fitness score of 4.2 by the end of training. The accuracy of these trained solutions on the testing data is also not impressive, achieving only a 23.9% correct classifications. By applying HyperNEAT as a feature learner and allowing backward propagation to train, both fitness score during training and classification correctness during testing increase to 5.7 and 58.4%, respectively. It is interesting to note that in both cases, the fitness score and the correct classifications are not completely correlated, that is, improvements in the fitness score can lead to decreases in classification accuracy.

Changing HyperNEAT to learn the weights of a CNN architecture, rather than the normal ANN architecture of HyperNEAT, does change performance (Figure 5). In this case, HyperNEAT by itself plateaus faster at a lower fitness level, achieving an average fitness score of 4.1 by the end of training, but the accuracy of these trained solutions on the testing data improves to 27.7% correct classifications. On the other hand, feature learning with HyperNEAT on a CNN architecture significantly improves both metrics, achieving a 7.0 fitness score and a 92.1% correct classifications. An example of the feature maps generated by these HyperNEAT champions can be seen in Figure 6.



(a) (b)

Figure 6. **Visualization of Feature Maps Generated by HyperNEAT.** Example feature maps are shown for the MNIST digit three (a) and an example container image from BCCT200 (b). Interestingly, there is a distinct pattern along the feature dimension (top to bottom), demonstrating patterns in the geometry of the feature maps.

5.2. BCCT200

Some recent work in image classification, specifically ship or vessel recognition from overhead satellite imagery, has been done using various traditional computer vision approaches [8] and ideas borrowed from compressive sensing [18]. This experiment compares HyperNEAT and traditional methodologies on the barge, cargo, container and tanker (BCCT200) ship recognition database created by Harguess et al. [18], which is transformed to a lower resolution (28×28 pixels) for this experiment. The BCCT200 data was split into 30 random permutations of training (75%) and testing (25%) datasets and the average result from all 30 trials is reported for the four top performers for this task from [7] along with a baseline result using principal components analysis (PCA) and nearest neighbors (NN). The HyperNEAT experiments were performed using HyperNEAT as a feature extractor with a CNN architecture, except that KNN was used as the classifier. The results are shown in Figure 7.

The best overall result was obtained using the raw pixels of the image with L_0 optimization, as described in [7], with an accuracy of 83.7%. PCA + NN provided a baseline result of 74.5% accuracy, while raw pixels with support vector machines (raw + SVM) and independent components analysis with L_0 optimization (ICA + L_0) performed at 83% and 80.6%, respectively. HyperNEAT for feature extraction with CNN architecture had an average of 65.3% with a peak performer at 68% accuracy. However, as seen in Figure 8, the accuracy is still steadily increasing with each generation. In lieu of time, the generations were halted at 200, however, the final accuracy after 2500 generations (similar to Figure 5) would likely be significantly higher. One clear implication of this is that while the traditional approaches might have a higher accuracy currently for classifying the BCCT200 data, their current implementations and parame-

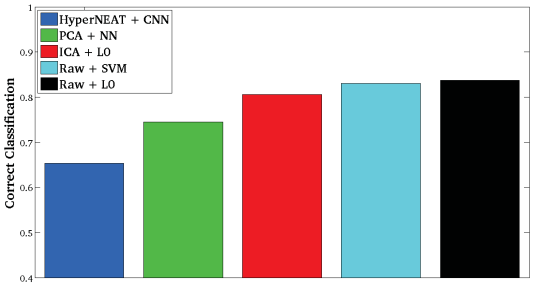


Figure 7. **HyperNEAT performance with CNN architecture as compared to traditional approaches on BCCT200.**

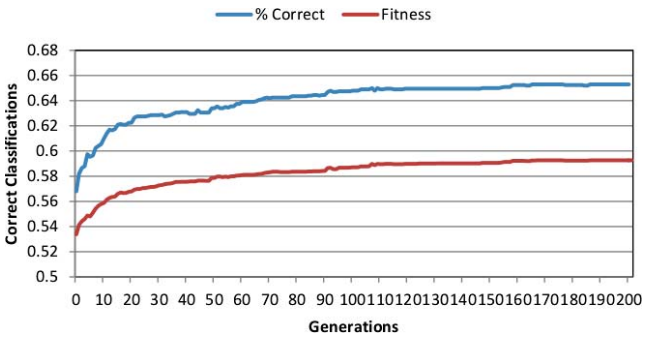


Figure 8. **Correct classifications per generation for HyperNEAT with CNN architecture for BCCT200.**

ter settings do not allow for additional gains in performance, whereas more generations and/or better strategies for classification have potential performance enhancement capabilities for the HyperNEAT architecture methodology.

6. Discussion and Future Work

The challenge for HyperNEAT is that, by training an indirect encoding, the ability to control precise weights is diminished, thereby creating difficulties in making fine adjustments for tasks such as classification. This challenge can be addressed by applying HyperNEAT as a feature learner for another ML approach that then makes the fine adjustments for the task, as shown in this paper. Indeed, the results in this paper demonstrate that HyperNEAT successfully learns features to train a backward propagation trained ANN. Interestingly, changing to the convolutional neural network architecture had a significant impact on HyperNEAT's performance. Two interesting questions arise: (1) Are there more effective architecture choices? and; (2) How can they be discovered? A path to answering these questions may be an extension of HyperNEAT known as Evolvable Substrate HyperNEAT (ES-HyperNEAT; [20]). In ES-HyperNEAT, the ANN substrate is not defined a priori (except for inputs and outputs); instead, the pattern generated by the CPPN determines the placement neurons in the hidden layers.

An exciting implication of this work is that evolution provides a means to learn a representation that is well-suited for the target machine learning approach. Through reinforcement learning, evolution can measure fitness as a function of how well an approach performs on the generated representation. Because the fitness measure does not depend on a particular error signal or gradient, multiple metrics can be incorporated. For example, the results in the paper incorporated several classification characteristics into the fitness function, such as measures of true positives, true negatives, false positives, and false negatives. By combining measures, evolution can explore multiple paths through the different metrics. In addition, the many candidate solutions that evolution generates will perform differently from each other and these solutions may be combined to create an enhanced feature set. Finally, because the fitness measure can be based upon performance of the trained solution on data not seen by ML approach, the discovered features may encourage learning that generalizes. Thus there exists an opportunity to encourage representations that encourage generalizability.

7. Conclusion

This paper investigated deep learning through NeuroEvolution. While NE has been limited in the size of ANNs that could be effectively trained in the past, novel algorithms that operate on indirect encodings, such as HyperNEAT, allow effective evolution of large ANNs. Prior work with HyperNEAT has shown promise in simple vision tasks that operate on a raw visual field, but with non-deep architectures. By itself, HyperNEAT struggles to find ANNs that perform well in image classification; however, HyperNEAT demonstrates an effective ability to act as a feature extractor by being combined with backward propagation. Thus HyperNEAT provides a potentially interesting path for combining reinforcement learning and supervised learning in image classification, as evolution and lifetime learning combine to create the capabilities in biological neural networks.

References

[1] T. Aaltonen et al. Measurement of the top quark mass with dilepton events selected using neuroevolution at CDF. *Physical Review Letters*, 2009.

[2] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

[3] O. J. Coleman. *Evolving Neural Networks for Visual Processing*. PhD thesis, The University of New South Wales, Kensington, Austrailia, 2010.

[4] D. B. D'Ambrosio and K. O. Stanley. Generative encoding for multiagent learning. In *Proceedings of the*

Genetic and Evolutionary Computation Conference (GECCO 2008), New York, NY, 2008. ACM Press.

[5] J. Gauci and K. O. Stanley. Generating large-scale neural networks through discovering geometric regularities. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 8, New York, NY, 2007. GECCO-2007, ACM.

[6] T. Gliga and G. Dehaene-Lambertz. Structural encoding of body and face in human infants and adults. *Journal of Cognitive Neuroscience*, 17(8), August 2005.

[7] J. Harguess*, S. Parameswaran*, K. Rainey*, and J. Stastny. Vessel classification in overhead satellite imagery using learned dictionaries. In *SPIE Optical Engineering+ Applications*, pages 84992F–84992F. International Society for Optics and Photonics, 2012.

[8] J. Harguess and K. Rainey. Are face recognition methods useful for classifying ships? In *Applied Imagery Pattern Recognition Workshop (AIPR), 2011 IEEE*, pages 1–7. IEEE, 2011.

[9] L. Hartwell, J. Hopfield, S. Leibler, and A. Murray. From molecular to modular cell biology. *Nature*, 402, 1999.

[10] M. Hausknecht, P. Khandelwal, R. Miikkulainen, and P. Stone. Hyperneat-gpp: A hyperneat-based atari general game player. In *Proceedings and the Genetic and Evolutionary Computation Conference*, page 8, Philadelphia, Pennsylvania, July 2012. ACM Press.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[12] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[13] D. H. Hubel and T. N. Wiesel. Receptive field of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591, 1959.

[14] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. McGraw-Hill, New York, fourth edition, 2000.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classfication with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[16] Q. V. Le, L. A. Isbell, J. Matsumoto, M. Nguyen, E. Hori, R. S. Maior, C. Tomaz, A. H. Tran, T. Ono, and H. Nishijo. Pulvinar neurons reveal neurobiological evidence of past selection for rapid detection of snakes. *Proceedings of the National Academy of Sciences*, 110(47), November 2013.

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

[18] K. Rainey and J. Stastny. Object recognition in ocean imagery using feature selection and compressive sensing. In *Applied Imagery Pattern Recognition Workshop (AIPR), 2011 IEEE*, pages 1–6. IEEE, 2011.

[19] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller. The manifold tangent classifer. In *Advances in Neural Information Processing Systems*, 2011.

[20] S. Risi and K. O. Stanley. An enhanced hypercube-based encoding for evolving the placement, density and connectivity of neurons. *Artificial Life*, 18(4):331–363, 2012.

[21] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

[22] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162, 2007.

[23] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15, 2009.

[24] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[25] P. Verbancsics and K. O. Stanley. Evolving static representations for task transfer. *Journal of Machine Learning Research*, 11, 2010.

[26] P. Verbancsics and K. O. Stanley. Constraining connectivity to encourage modularity in hyperneat. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, New York, NY, 2011. ACM Press.

[27] G. Wagner and L. Altenberg. Complex adapatations and the evolution of evolovability. *Evolution*, 50, 1996.

[28] S. Whiteson. Improving reinforcement learning function approximators via neuroevolution. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1386–1386, New York, NY, USA, 2005. ACM.

[29] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[30] M. J. Zigmond, F. E. Bloom, S. C. Landis, J. L. Roberts, and L. R. Squire, editors. *Fundamental Neuroscience*. Academic Press, London, 1999.