

Evolving the Starting State of Autoencoders

Midterm Project Report

Christopher Ross

Jonathan Brant

Zak Roessler

1. INTRODUCTION

Image recognition and classification problems have long captured the interest of computer vision researchers [1, 2, 3]. The ability to process and accurately identify visual input has a vast array of interesting applications across the field of AI and other scientific disciplines. Unsurprisingly, however, such an undertaking has proven to be a rather ambitious, with one of the primary obstacles being that of scale [4].

Perhaps the most straightforward method of processing images is to consider each pixel as a separate input into an artificial neural network (ANN) classifier, with the image resolution defining the distinct number of pixels. As image resolution increases, however, the number of classifier inputs will similarly grow exponentially, easily reaching inputs in the millions. Considering each pixel separately quickly becomes an inefficient approach and necessitates a method for dimensionality reduction.

In order to tackle such large problems, this project proposes the use of an evolved autoencoder. Unlike traditional autoencoders wherein the hidden layers (i.e. compressed feature vectors) are specified a priori, the research presented herein uses HyperNEAT to evolve connective CPPNs which in turn produces autoencoders that are, for all practical purposes, scale invariant and whose weights are decided in a more principled manner.

2. BACKGROUND

The fundamentals of traditional autoencoders are introduced along with the relevant details of the HyperNEAT algorithm.

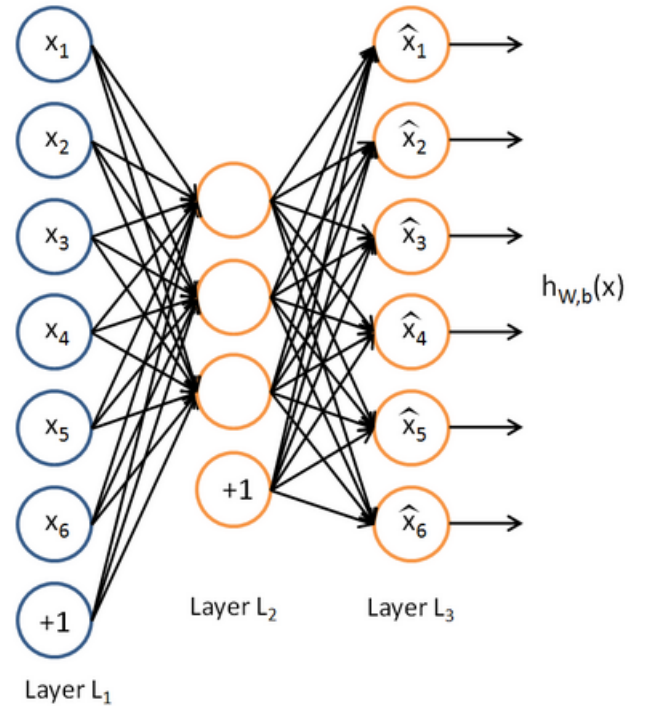
2.1 Autoencoders

Autoencoders are artificial neural networks that perform a non-linear form of principal components analysis (PCA) and attempt to extract the most salient features from the visible (input) layer [5, 6]. A simple autoencoder has a separate input/output pair for each "visible" feature of a domain

as well as a fully connected hidden layer that learns the most salient features of the uncompressed features in the visible layer. More generally, input is being encoded into a compressed representation and subsequently reconstructed. Figure 1 depicts the structure and encode/decode phase of a simple autoencoder.

After the compressed feature vector outputs have been decoded into the full feature set, a reconstruction error is calculated and used to determine the representational efficiency/accuracy of the hidden layer. The reconstruction error for each visible layer node is fed to a supervised training algorithm, most commonly backpropagation, which updates the weights of the autoencoder in a direction that minimizes reconstruction error [7].

Figure 1: An autoencoder with 6 visible units and 3 hidden units. The input is encoded from Layer L1 to Layer L2, and decoded from Layer L2 to Layer L3 [7].



2.2 NEAT

In order to automatically determine the autoencoder’s hidden layer topology, techniques are borrowed from the field of neuroevolution. One of the most successful and widely-used genetic algorithms for ANN evolution is the NeuroEvolution of Augmenting Topologies (NEAT). Three primary components separated NEAT from algorithms that preceded it: growth from minimal structure, speciation, and perhaps most importantly, historical markings [8].

Prior to NEAT, many Topology and Weight Evolving Artificial Neural Networks (TWEANNs) began evolution in an already high-dimensional state, leading to slow computationally intensive evolution and often poor results. NEAT addresses this issue by starting with minimal structure (no hidden nodes) and complexifying only when doing so positively impacts objective performance.

A potential problem with starting minimally, however, is that such structure will be optimized over time, often outperforming new structural innovations that haven’t yet had a chance to be fine-tuned. NEAT’s speciation component avoids premature penalization of new innovations by imposing explicit fitness sharing. This allows an ”adjusted fitness” to be calculated for each genome relative to other structurally similar genomes in its species, thus avoiding premature convergence and protecting new innovations.

Finally, at the core of NEAT are historical markings that uniquely identify each structural innovation. Historical markings solve the topology matching problem, allowing crossover operations without expensive topological analysis. Speciation also uses historical markings to determine the species in which to place each genome.

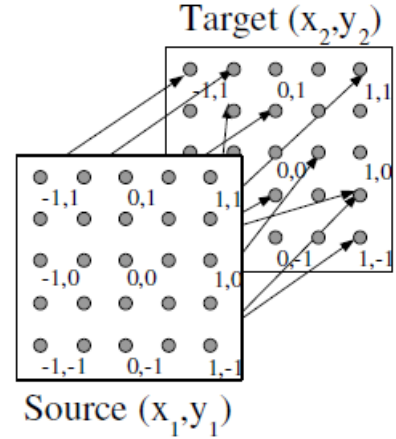
NEAT has been proven to be a highly successful algorithm for control and sequential decision tasks in relatively low-dimensional domains; however, as a direct encoding, the NEAT genotype contains a separate gene for each structural component. For high-dimensional domains, this can quickly become unwieldy and difficult to optimize. Moreover, as NEAT considers every input separately, it has no concept of geometric regularities making it a less than optimal choice for domains with a strong visual component.

2.3 HyperNEAT

Given the complexity of the domain under consideration and the strong geometric component, the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) is a natural choice of algorithms [9].

As an indirect encoding, HyperNEAT does not impose a one-to-one mapping between genes in the genotype and structural components (i.e. connections and neurons in the case of ANNs) in the phenotype. Instead, HyperNEAT evolves Compositional Pattern Producing Networks (CPPNs) that describe connectivity patterns as a composition of canonical functions [10, 9]. The evolved CPPN queries every possible connection within a $2n$ -dimensional substrate and outputs the weight of the connection between those two points, effectively drawing an n -dimensional connectivity pattern which is then interpreted as an ANN and evaluated in the given domain.

Figure 2: The ”state-space sandwich” substrate - ideal for visual mapping [9].



The substrate configuration is hand-designed in order to accurately represent the geometry of the task-domain. Among the many possible substrate configurations, one of the most widely used is the ”state-space sandwich” substrate wherein a source sheet of neurons is fully connected to a target sheet, as shown in Figure 2. A modified version of the state-space sandwich will be used to represent the visible and hidden layers of the desired autoencoder. HyperNEAT’s ability to efficiently generate extremely large scale networks that exhibit virtually no loss of function while using a highly compressed representation that accounts for domain geometry makes it an ideal candidate for the image image processing tasks considered in this paper.

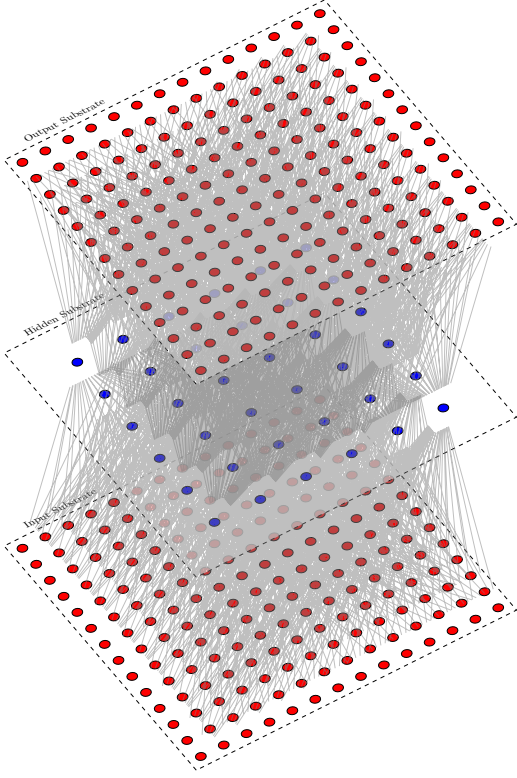
3. APPROACH

While autoencoders have been considered viable methods for data compression or dimensionality reduction since their introduction in 1988 [5], they have, to the author’s knowledge, always been statically constructed such that their hidden layer is fully specified a priori. This paper seeks to evaluate the potential of leveraging HyperNEAT for the evolution of autoencoders. Results of such an approach will be analyzed both quantitatively (i.e. the efficacy of evolved connection weights toward reducing reconstruction error) and qualitatively (i.e. how closely does the reconstructed image resemble the original).

The substrate configuration should be chosen to match that of a typical autoencoder structure in order to capture the encode/decode concept. This includes a two-dimensional sheet of neurons representing each pixel in the given image, a hidden layer sheet with fewer (typically half) neurons, which will in turn be connected to a sheet the same size as the input sheet. The ”hidden layer” sheet effectively represents the hidden layer of a traditional autoencoder. This ”stacked” state-space sandwich is depicted in figure 3.

A HyperNEAT-evolved CPPN will query each possible connection between the input sheet and the hidden sheet as well as between the hidden sheet and the output sheet, in order to determine the weight and existence of every possible con-

Figure 3: The “state-space sandwich” substrate used in the evolved autoencoder experiments. The hidden layer may vary depending on the experiment configuration, but the general three-tiered structure will remain.

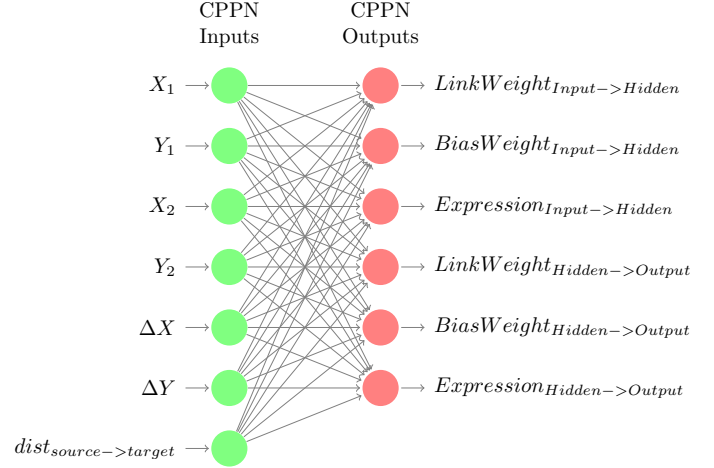


nection.

Evolution will begin with a minimally connected CPPN (no hidden nodes) with seven input nodes and six output nodes. The input nodes accept the Cartesian coordinates (x and y position values) for each line segment endpoint, the deltas between the x and y components for each connection endpoint, and the euclidean distance between the source and target nodes. The output nodes express the connection weight, bias weight, and the link expression output (LEO) indicator from both the input sheet to the hidden sheet and the hidden sheet to the output sheet (outputs are duplicated for each connection layer). One such minimal CPPN is shown in figure 4.

The resulting connectivity pattern will be interpreted as an autoencoder and will be both trained and evaluated on samples from the Mixed National Institute of Standards and Technology (MNIST) handwritten digit dataset [11], which is often used as a benchmark dataset for image classification problems. The handwritten digits in the MNIST dataset are greyscale and each pixel is encoded as an integer in the range 0 to 255 (which represents the intensity of the greyscale pixel). Reconstruction error for each sample image will be computed based on the difference between the output pixel greyscale intensity compared to its respective input pixel intensity.

Figure 4: The minimal, starting-state (no hidden nodes) CPPN. The inputs to the CPPN include the X and Y coordinates on the source substrate, the X and Y coordinates on the target substrate, the delta between the X-components and Y-components, and the euclidean distance between the source and target points. If the CPPN is querying the connections between the input and hidden sheets, it will output (for every combination of points) the connection weight between the sheets, the weight from the bias to the target sheet, and the link expression (i.e. whether the connection will be expressed in the phenotype). Similarly, if the CPPN is querying the connections between the hidden and output sheets, it will output the same for every combination of points.



The training process will use traditional backpropagation, modifying the autoencoder weights in order to reduce reconstruction error. After backpropagation has been run for the specified number of iterations (which will vary based on the experiment configuration), the overall reconstruction error will be taken as the score of the CPPN that generated the autoencoder under evaluation. The maximum fitness attainable is equivalent to the product of the number of validation samples and their resolution (i.e. setting the correct greyscale intensity for each pixel in every validation sample). Therefore, the fitness score for a given autoencoder evaluation is calculated by subtracting the sum of the reconstruction errors for all validation samples from the maximum fitness, and then dividing by the maximum fitness. Algorithm 1 describes the full process.

In this way, HyperNEAT is generating the autoencoder starting state and backpropagation is fine tuning it. The hope is that HyperNEAT will generate a structure that is easily trained to a point of minimal reconstruction error.

4. DISCUSSION OF CURRENT STATE

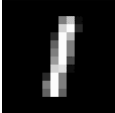
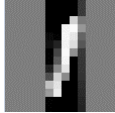


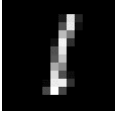
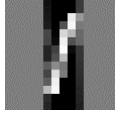
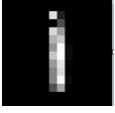
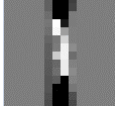
The current state discussion includes the results observed thus far from initial experimentation, an analysis of those results (including alternate strategies in the event that experiment outcomes were poor), and final experiments that are planned to bring the project to a hopefully informative completion.

Algorithm 1 Autoencoder Training/Evaluation Algorithm

```
1: procedure EVALUATEAUTOENCODER( $s, res$ ) ▷ Scores autoencoders on  $s$  image samples at  $res$  resolution
2:    $max\_fitness \leftarrow s * res$  ▷ Maximum fitness is product of # of image samples and their resolution
3:   for Each genome in population do
4:     for Each training image do
5:       while  $iter < n$  do ▷ Backpropagate for  $n$  iterations
6:         Activate network
7:         Compute error and update weights
8:       end while
9:     end for
10:     $error\_sum \leftarrow 0$  ▷ Initialize the sum of errors
11:    for Each validation image do
12:      for Image pixel do
13:         $error\_sum := error\_sum + abs(input\_value - output\_value)$  ▷ Error is magnitude of difference between
        input and output pixel intensity
14:      end for
15:      Compute network error in one activation (no backpropagation)
16:    end for
17:     $fitness := error\_sum + (max\_fitness - error\_sum) / max\_fitness * 100$ 
18:  end for
19: end procedure
```

Recall that results will be analyzed from both a quantitative and qualitative standpoint. Quantitative analysis is concerned with the final reconstruction error of the networks after evolution and backpropagation while the qualitative analysis is based on a subjective assessment of how closely the reconstructed image matches the original. Table 3 enumerates the experiment parameters used in both the NEAT and HyperNEAT experiments presented in this section while table 2 lists the NEAT-specific evolution parameters.

Table 1: HyperNEAT Qualitative Results

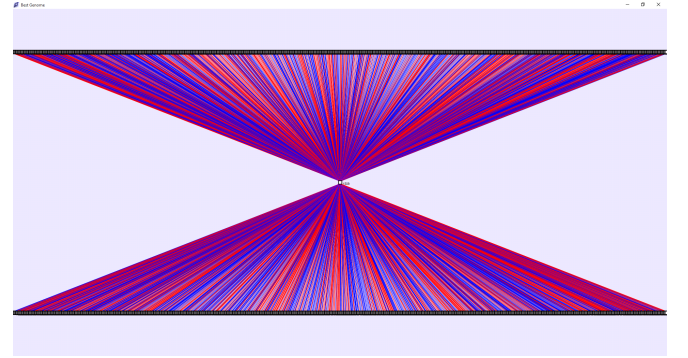
Source	Reconstructed	Set	Index
		Training	30
		Training	60
		Validation	85
		Validation	95

The original images were scaled down from 28x28 pixels (greyscale) to 14x14 for training efficiency. In order to evaluate the qualitative results, both source and reconstructed images were written to bitmap files and scaled to 800% (so that they're easier to view).

4.1 NEAT Results

During the initial stages of experimentation, standard NEAT was used to directly evolve autoencoders. The result was slow evolution (due to the massive search space) and poor reconstruction performance. Figure 5 depicts the starting state of a NEAT-evolved autoencoder, with inputs fully connected to a single hidden node, and that hidden node fully connected to the output nodes. The evolved network topology after 1,000 generations is shown in figure 6, with the nonlinearities introduced by the additional hidden layer structure aiding in the reduction of training error (though not in the manner hoped for, as described below).

Figure 5: The starting state of a NEAT-evolved autoencoder.



Quantitative reconstruction accuracy rarely exceeded 60% accuracy, after which improvements were extremely slow due to consistently getting stuck in local optima. Due to the poor reconstruction performance, qualitative results were not closely analyzed because the reconstructed image would have been fairly amorphous and unrecognizable.

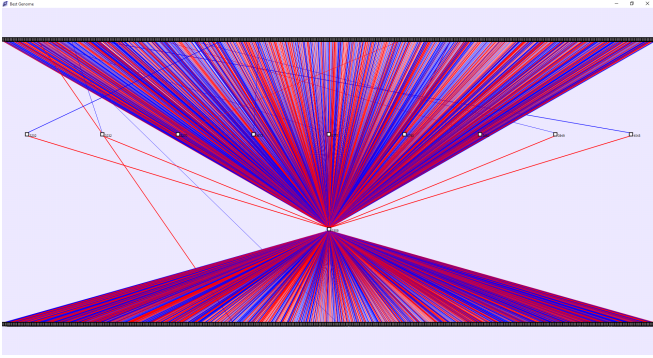
However, the deeper problems was that NEAT was unable to generate a viable autoencoder starting state due to its inability to account for task geometry and the vast dimensionality

Table 2: Configuration used for Initial NEAT and HyperNEAT experiments

Parameter	Value	Description
Training Sample Proportion	80%	Percentage of sample images used for training
Number of Backpropagation Iterations	1	Number of backpropagation iterations performed during training
Learning Rate	0.01	Learning Rate (to control backpropagation convergence speed)
Image Resolution Reduction Factor	2	Reduction factor of source image (i.e. a 28x28 image becomes 14x14 with a reduction factor of 2)
MNIST Digit	1	MNIST handwritten digit dataset image used

Table 3: NEAT Parameters

Parameter Name	Parameter Value
Population Size	250
Connection Proportion	0.05
Number of Species	10
Crossover Probability	0.5
Interspecies Mating Rate	0.001
Mutate Link Weights Probability	0.9
Mutate Add Connection Probability	0.1
Mutate Add Node Probability	0.1
Mutate Delete Connection Probability	0.001

Figure 6: A NEAT-evolved autoencoder after 1,000 generations of evolution. Structure has been added in order to aid in reducing training error, but the qualitative results were not quite as expected, resulting in mostly black images.

of its search space. It was primarily these realizations that led to the incorporation of HyperNEAT.

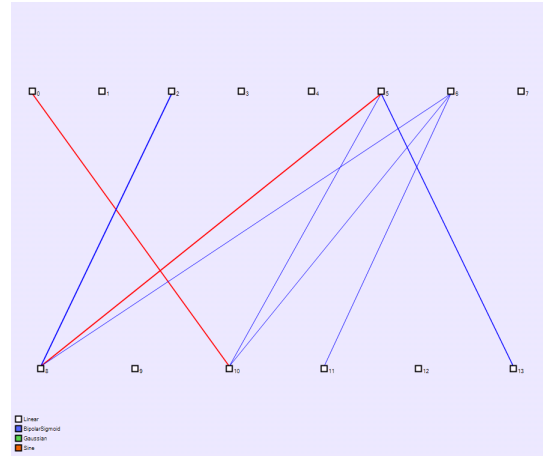
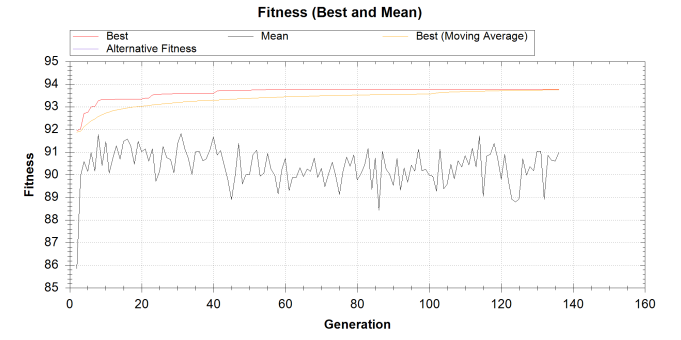
4.2 HyperNEAT Results

After the fairly underwhelming results with NEAT, HyperNEAT was incorporated in the hopes that incorporating information about the domain geometry would boost qualitative performance. Figure 4 depicts a fully connected, minimal CPPN (i.e. no hidden nodes) of the form that was used in the below experiments; however, most runs started with a CPPN that was 5% connected and connections were added as they were found beneficial with regard to reducing training error.

Figure 7 depicts such a CPPN, which was evolved in 136 generations and had a fitness of 0.937 (93.7% accuracy). Note that the trial that produced this CPPN ran for 136 generations, but it exhibited a very steep increase in fitness in just the first couple of generations as shown in figure 8. This may be due to the fine-tuning of backpropagation and the highly

specific manner in which it rewards networks that are more easily tunable; however, more experimentation is needed in order to determine whether such an assertion is merited.

TODO: ADD PROOF THAT NODE ACTIVATIONS ARE CHOSEN PROBABILISTICALLY AND FUNCTIONS ARE INHERITED (not sure how to do the latter)

Figure 7: A CPPN evolved in 136 generations with a fitness of 0.937 (i.e. 93.7% reconstruction accuracy).**Figure 8:** Fitness of an evolved CPPN over 136 generations. Notice the sharp increase with the first couple of generations followed by relatively stable dynamics.

Interestingly, quantitative results for HyperNEAT were quite similar to that of traditional NEAT, with networks consistently training to about 90% accuracy (given the experimental configuration shown in table 3). However, the manner in which those scores were reached differed considerably. While NEAT reached such an error by simply turning all pixels black, HyperNEAT managed to produce results that

were qualitatively similar to the original. The figures in table 1 depict a sample of the source image and reconstructed images, giving a sense of the autoencoders' qualitative performance.

4.3 Plan for Completion

The initial experiments consisted only of two digits from the MNIST dataset using a similarly limited parameter set. Future experiments will include the following:

Training/Validation Proportion: The proportion of the image sample set used for training versus validation will be varied in order to get an idea of the minimum training sample size required in order to produce acceptable results.

Maximum Backpropagation Iterations: The number of iterations of backpropagation that is run will be both increased and decreased in order to determine the right balance between evolution of the autoencoder starting state and the extent to which it needs to be trained following evolution. Perhaps the most interesting question here is whether evolution can produce robust autoencoders with minimal intervention from gradient-based optimization.

Image scaling: Currently, experiments are being run with images that are half the resolution of the original. These images will be increased to full resolution (28x28 pixels) in order to confirm the ability of HyperNEAT-evolved autoencoders to scale.

MNIST Generalization: The experiments run thus far have been limited to the first two digits of the MNIST dataset. Higher order digits vary significantly in geometric structure and will be a good indication of the evolved autoencoder's ability to reconstruct a wider range of shapes.

5. REFERENCES

- [1] L. Shapiro, *Computer vision*. Upper Saddle River, NJ: Prentice Hall, 2001.
- [2] T. Morris, *Computer vision and image processing*. Basingstoke: Palgrave Macmillan, 2004.
- [3] M. Sonka, *Image processing, analysis, and machine vision*. Toronto: Thompson Learning, 2008.
- [4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. [Online]. Available: <http://arxiv.org/abs/1409.0575>
- [5] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics*, vol. 59, no. 4-5, pp. 291-294, 1988. [Online]. Available: <http://dx.doi.org/10.1007/BF00332918>
- [6] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," *Unsupervised and Transfer Learning Challenges in Machine Learning, Volume 7*, p. 43, 2012.
- [7] S. U. Computer Science Department. (2013) Autoencoders and sparsity. [Online]. Available: http://ufldl.stanford.edu/wiki/index.php/Autoencoders_and_Spa
- [8] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99-127, Jun. 2002. [Online]. Available: <http://dx.doi.org/10.1162/106365602320169811>
- [9] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artif. Life*, vol. 15, no. 2, pp. 185-212, Apr. 2009. [Online]. Available: <http://dx.doi.org/10.1162/artl.2009.15.2.15202>
- [10] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131-162, Jun. 2007. [Online]. Available: <http://dx.doi.org/10.1007/s10710-007-9028-8>
- [11] J. H. U. Center for Imaging Science. (2005) Handwritten digit database. [Online]. Available: <http://cis.jhu.edu/~sachin/digit/digit.html>