AN EVOLUTIONARY APPROXIMATION TO CONTRASTIVE DIVERGENCE
IN CONVOLUTIONAL RESTRICTED BOLTZMANN MACHINES

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science

By

RYAN MCCOPPIN
B.S., Wright State University, 2012

2014
Wright State University

______________________________
Mateen Rizki, Ph.D.
Thesis Director

______________________________
Mateen Rizki, Ph.D., Chair
Committee on Final Examination
Department of Computer Science and Engineering
College of Engineering and Computer Science

Mateen Rizki, Ph.D.

Micheal Raymer, Ph. D.

John Gallagher, Ph. D.

Robert E. W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School
ABSTRACT

McCoppin, Ryan. M.S. Department of Computer Science and Engineering, Wright State University, 2014. An Evolutionary Approximation to Contrastive Divergence in Convolutional Restricted Boltzmann Machines

Deep learning is an emerging area in machine learning that exploits multi-layered neural networks to extract invariant relationships from large data sets. Deep learning uses layers of non-linear transformations to represent data in abstract and discrete forms. Several different architectures have been developed over the past few years specifically to process images including the Convolutional Restricted Boltzmann Machine. The Boltzmann Machine is trained using contrastive divergence, a depth-first gradient based training algorithm. Gradient based training methods have no guarantee of reaching an optimal solution and tend to search a limited region of the solution space. In this thesis, we present an alternative method for synthesizing deep networks using evolutionary algorithms. This is a breadth-first stochastic search process that utilizes reconstruction error along with additional properties to encourage evolution of unique features. Using this technique, potentially a larger region of the solution space is explored allowing identification of different types of solutions using less training data. The process of developing this method is discussed along with its potential as a viable replacement to contrastive divergence.
Table of Contents

1. Introduction .................................................................................................................. 1
2. Background .................................................................................................................... 5
   2.1 Restricted Boltzmann Machine ............................................................................... 6
   2.2 Convolutional Restrictive Boltzmann Machines .................................................... 12
3. Evolutionary Learning for Training Deep Networks ..................................................... 16
   3.1 Background on Evolution Learning ....................................................................... 17
4. System Description and Experimental Design ............................................................... 19
   4.1 System Description ............................................................................................... 19
   4.2 CRBM System Design ......................................................................................... 23
   4.3 Evolutionary System Design ............................................................................... 25
5. Experimental Results .................................................................................................... 36
6. Conclusions .................................................................................................................. 43
7. Future Works ............................................................................................................... 46
References .......................................................................................................................... 47
List of Figures

Figure 1 Example of CRBM weights matrices. These matrices were trained on the MNIST handwritten character dataset [10].......................................................................................................................... 3
Figure 2 CRBM Feature (left), Genome representation (middle), Phenome feature (right)........ 4
Figure 3 A Deep Learning Network...................................................................................................................... 5
Figure 4 A RBM model ......................................................................................................................................... 6
Figure 5 RBM Network (left), Gibbs' Sampling Process (right) ........................................................................ 10
Figure 6 A Data Sample, An Identity-Like Weight Transform and a Possible Hidden Output ........ 21
Figure 7 Weights (top left), Expectation (bottom left), Faint Reconstruction (top right) ................. 22
Figure 8 The Original Image, One Feature Used, Hidden Activated Regions, and Reconstruction23
Figure 9 The Binary Weight Matrices - Bxy........................................................................................................... 25
Figure 10 Evolutionary Cycle................................................................................................................................. 27
Figure 11 Mutation Operator Examples .................................................................................................................. 29
Figure 12 Gaussian and Dilation Mutations .......................................................................................................... 30
Figure 13 Crossover Diagram................................................................................................................................. 30
Figure 14 Process of New Generations.................................................................................................................... 31
Figure 15 Examples from the MNIST Dataset [10]............................................................................................. 32
Figure 16 Learning with 20 Images (time vs error).................................................................................................. 37
Figure 17 Weights from ES (above), Weights from CRBM (below) ................................................................. 38
Figure 18 Evolutionary System Reconstructions .................................................................................................. 38
Figure 19 Evolutionary System Final Population of 50 Members ........................................................................ 39
Figure 20 Learning with Batches of Images (Time vs Error) ............................................................................. 40
Figure 21 Weight Comparison – CRBM (top), Evolutionary (bottom) ......................................................... 41
Figure 22 Constitution of a Feature ....................................................................................................................... 41
Figure 23 CRBM Reconstructions of Digits ........................................................................................................... 42
Figure 24 Fitness Values per Iteration ..................................................................................................................... 42
Figure 25 Samples through Stages of Evolution ................................................................................................. 43
Figure 26 Thick or Thin Images ............................................................................................................................. 44
Figure 27 Repeated Feature in Two Regions of Weight Space ......................................................................... 45
1. Introduction

In recent years there has been great interest in the field of deep learning, spurred by the work of Geoff Hinton (1) who demonstrated deep learning is a viable approach to extract features that effectively classify large, multi-class data sets. Deep learning is a subfield of machine learning that extends many of the concepts and techniques developed to synthesize a multilayered neural network architecture, layer-by-layer, such that each layer provides a greater level of abstraction.

There are many approaches to training multi-layered networks, one of which is to use a gradient descent algorithm to adjust the weights of each fixed layer using backpropagation. The problem with the use of backpropagation for training networks is the effect of the error signal dissipates as the depth of the network increases. Deep learning attempts to exploit gradient descent learning using a greedy approach that trains the network bottom-up, layer-by-layer.

Gradient descent is a greedy, depth-first search for an optimal solution that is well suited to problems having a single optimal solution positioned in a search space such that a gradient tends to direct the search toward the optimal solution. Historically, learning techniques based on greedy, depth first search exhibit difficulties when applied to problems with multiple local optima or search spaces lack consistent gradients. To compensate for some of these difficulties, Hinton developed a stochastic gradient descent algorithm called contrastive divergence to train restricted Boltzmann machines (RBM) (2). While still essentially a depth-first algorithm, contrastive divergence is not a greedy search, so it is able to escape local minima with some degree of success with large amounts of training data. In addition, special control parameters such as momentum and decay are typically implemented which allow the user to adjust the training process to mitigate the problems associated with local minima.

These new deep learning techniques have been recognized in recent years for their ability to learn features useful for classification in several important problem domains (3). For instance, LeCunn has
developed convolutional neural networks that have achieved remarkable classification accuracy on large image datasets (4). These successes have encouraged many researchers to explore variations of deep learning to attempt to develop even more robust classification techniques.

With all of its successes there are still issues with deep learning. It is limited in the amount of the search space it can explore precisely because of its depth-first search strategy. While it can escape some local minima, much of the search space remains unexplored. Another more troubling requirement for deep learning is its dependence on the availability of large amounts of training data, without which generalization of a probability distribution is not possible. This makes it difficult to apply deep learning to many problems with limited access to data. Some techniques have been developed to overcome the lack of data including data distortion and the use of simulated data (5). Lastly, many deep learning techniques are computationally expensive. Over the years many solutions have been proposed to accelerate the learning process by the use of multiprocessing (e.g. GPU) (6) and more efficient training techniques that train the network architecture using a layer-by-layer unsupervised learning algorithm (7).

This thesis explores an alternative approach to training RBMs, specifically, using evolutionary learning techniques to explore the solution space in a breadth-first search. This technique allows the search process to more readily escape local optima and is no longer dependent on learning a probability distribution of the data. This permits the learning system to use less training data when developing a solution. It is possible that an evolutionary-based system could also be trained faster by not having to compute gradients and by exploiting its highly distributive structure to use high performance computer systems. The goal of this thesis is to develop a learning technique that produces solutions of comparable quality to the gradient descent techniques while requiring less training data.

Machine learning is an important branch of computer science because it facilitates learning representation and generalization of concepts from data including important applications such as classification, grouping/clustering, dimensionality reduction, data generation, and anomaly detection.
Over the past ten years there have been many advances in the field of machine learning especially the subfield of deep learning. Deep learning is the concept of using layers of non-linear transformations to represent data in a more abstract and discrete form. Several different architectures have been developed over the past few years in order to better achieve this result (8). One of those architectures is the restricted Boltzmann machine developed by Geoff Hinton (1). His work helped create a new area of generative models some of which are applied as convolutions of images. Convolutional based RBM (9) networks are of special interest because of their ability to process large images. These networks are interesting because you can train each layer independently to learn an abstract representation of the data and the resulting network has a generative property that allows reconstruction of the input data.

**Figure 1 Example of CRBM weights matrices. These matrices were trained on the MNIST handwritten character dataset [10].**

Training Convolutional RBMs using contrastive divergence, a stochastic gradient descent technique, is an effective method to train a network to model data. The network being trained is represented by a collection of two-dimensional matrices (Figure 1). The use of CD can be a costly process because the training process uses computationally costly gradients and there are many tuning parameters such as a learning rate that must be carefully adjusted to converge on a solution. For example, to train a network to model a complex data set, the learning rate must be adjusted to balance the need for small changes in the weight at the expense of many iterations of the CD algorithm. Unfortunately there is no guarantee that we can or will reach an optimum solution so the process of tuning parameters is simply trial and error. An alternative to this method is to use an evolutionary algorithm to learn the transformation parameters for
the model. Although evolutionary techniques do not guarantee reaching an optimum, the breadth of the search and the ability to backtrack allows for many regions of the optimization surface to be explored.

This may allow us to achieve a better solution using less training time. Instead of minimizing the gradient, we choose to minimize the reconstruction error. The reconstruction error is the distance between a data sample and its reconstructed representation derived from the hidden space. This is more of a direct approach to develop weights that minimize the reconstruction error and avoids the gradient computation altogether. In order to get an accurate estimate of the fitness, the reconstruction error must be computed over many images.

Composing the transformation as a summation of Gaussian filters significantly reduces the parameter space required to search. We use domain information to help guide the search process as well; for instance, increasing the chances of Gaussians being placed in the same region of the weight space. For each feature developed by the Convolutional RBM (shown in Fig. 2 on the left), a binary substitute is developed (middle) and Gaussian kernels are placed on the on locations through a summation of Gaussian filters (right).

![Image](image.png)

**Figure 2 CRBM Feature (left), Genome Representation (middle), Phenome Feature (right)**

The goal of this thesis is to show a set of features can be developed in the CRBM model using a binary genome instead of real valued weights developed by gradient descent. This method may save time by avoiding the computation of the gradient.
2. Background

From the early days of machine learning the goal has been the same. We want computers to be able to understand high level thoughts and ideas, especially the brain. Rosenblatt was one of the earliest researchers in this area who attempted to mimic the computational processing of a single brain cell. He titled his model a “perceptron”. Although the original Perceptron consisted of a single layered network of linear processing elements, these ideas were quickly extended to multiple layered networks using combinations of both nonlinear and linear processing elements. These multilayered networks take data vectors and map them to a class label using the concept of backpropagation to train the network. Researchers attempted to build deeper networks by stacking groups of these “neurons” in layers. This allowed for more abstract relationships to be formed among the higher layer neurons. It quickly became clear that there were limits to this approach when it was noticed (experimentally) that the gradient “vanishes” as the number of layers increased (11). More complex learning techniques were needed to train deep multilayered networks.

![Figure 3 A Deep Learning Network](image)

The concept of deep learning emerged as an alternative training method to backpropagation. Some deep learning methods allow the gradient to be persevered to greater depths while others permit each layer to
be trained independently of the others. The idea of deep learning is illustrated in Figure 3 where data is being abstracted through consecutive layers of weights. Each successive layer is an abstract representation of the original data. One technique that allowed each layer to be trained independently is called the Restricted Boltzmann Machine (RBM).

2.1 Restricted Boltzmann Machine
An RBM is a stochastic bipartite network that learns the probability distribution of a given training set generated by a summation of features. The network is composed of visible units and hidden units. The visible units represent (model) the probability distribution of the data while the hidden units represent an abstraction of the data that can reliably reconstruct the original input. Since the connection between the visible and hidden units form a bipartite graph, there exists a tractable learning algorithm that trains a non-linear transformation function between these spaces (1).

The transformation function is represented by the set of edges in the graph as well as two bias terms used to offset the transformation from zero shown in Figure 4. The parameters of the transformation are determined by minimizing an energy term over the training set. As the energy decreases, the probability that a generated visible value is represented by the data distribution increases. How best to learn this transformation function is what is of interest in this thesis. Traditionally, the following energy function (eq 1) is minimized in order to maximize the probability of the data as shown in (eq 2) where v is the visible
data, \( h \) is the hidden representation, \( c, b \), are bias terms and \( W \) is the weight parameters. Also, \( c \) is a scalar term, \( h \) is a one-dimensional vector and \( W \) is a two-dimensional vector.

\[
E(v, h) = -c'v - b'h - h'Wv
\]  

(1)

The probability of the data (eq. 2) is formed as a function of the energy and normalization constant. If \( p(v) \) can be maximized for all samples in a dataset, it is said to be the probability distribution of that dataset.

The normalization constant \( Z \) causes the result to be a probability of \( v \) occurring in the data.

\[
p(v) = \sum_h \frac{e^{-E(v,h)}}{Z}, \text{ and } Z = \sum_v e^{-E(v,h)}
\]  

(2)

In a RBM model, \( p(v) \) is maximized through a gradient descent approximation. This is equivalent to minimizing a negative log loss function (eq 3) over the training data \( D \) where the parameters \( \theta = (b, c, W) \) are from the energy function and the size of the training set is \( N \) data samples.

\[
l(\theta, D) = -\frac{1}{N} \sum_{v \in D} \log p(v)
\]  

(3)

In order to simplify the analytical computation of the derivative, the concept of free energy (1) is used.

Free energy (eq 4) is a term borrowed from the field of physics and is defined as:

\[
F(v) = -\log \sum_h e^{-E(v, h)}
\]  

(4)

Free energy is used to remove the summation \( p(v) \) through the following derivation (eq 5) and updating the normalization term.
This new form for $p(v)$ is solely to make the calculation of the gradient easier. Remember that the goal is to maximize the probability of a data vector in terms of the parameters, so we find the gradient of the likelihood equation for a single point in (eq 6) using our new representation of $p(v)$ developed in (eq 5).

$$\begin{align*}
p(v) &= \frac{\sum_h e^{-E(v_h)}}{Z} = \frac{\sum_h e^{-E(v_h)}}{\sum_v \sum_h e^{-E(v_h)}} = \frac{e^{\log \sum_h e^{-E(v_h)}}}{\sum_v e^{\log \sum_h e^{-E(v_h)}}} = \frac{e^{-F(v)}}{\sum_v e^{-F(v)}} = \frac{e^{-F(v)}}{Z_F}, \text{where } Z_F \\
&= \sum_v e^{-F(v)}
\end{align*}$$

The gradient is now relatively simple to calculate by using the free energy. The first term of equation (eq 6) is called the positive phase (1); the positive phase moves the model toward the probability of the training data. The second term of the model is called the negative phase (1); it moves the model away from the inherent distribution of samples generated by the model which is initialized by random weight generation. Now all that needs computed is the gradient of the free energy term. The gradient can be computed in two steps. In (eq 7), the positive phase is computed and then used in the computation of the expected value (eq 8) for the negative phase.

$$\begin{align*}
-\frac{\partial \log p(v)}{\partial \theta} &= -\frac{\partial \log e^{-F(v)}}{\partial \theta} + \frac{\partial \log Z_F}{\partial \theta} = \frac{\partial F(v)}{\partial \theta} + \frac{\partial \log \sum_v e^{-F(v)}}{\partial \theta} \\
&= \frac{\partial F(v)}{\partial \theta} - \frac{\sum_v \frac{\partial F(v)}{\partial \theta} e^{-F(v)}}{\sum_v e^{-F(v)}} = \frac{\partial F(v)}{\partial \theta} - \sum_v p(v) \frac{\partial F(v)}{\partial \theta} \\
&= \frac{\partial F(v)}{\partial \theta} - E_v \left[ \frac{\partial F(v)}{\partial \theta} \right]
\end{align*}$$
Ideally, these terms should be used in practice but notice the negative term consists of $p(v,h)$ which is intractable to compute because it requires computing the expectation over all possible $v$ and $h$. The positive term can be calculated directly from the data while the negative term becomes complicated because of its dependency on $p(v)$ which is unknown. This problem is solved using Gibbs sampling (12) to approximate $p(v,h)$ by iteratively sampling from $v$ and $h$ using their respective conditional distributions: $p(v|h)$ and $p(h|v)$. Sampling between these distributions will converge to $p(v,h)$. This is possible because of the bipartite nature of the model which makes $p(v|h)$ and $p(h|v)$ independent and easy to calculate for a sample. Each of these terms is computed as following:

$$
p(v_i = 1|h) = \frac{p(v_i = 1, h)}{p(h_i = 1)} = \frac{e^{-E(v,h)}}{Z} \frac{\sum_v e^{-E(v,h)}}{\sum_v e^{-E(v,h)}} = e^{-E(v,h)} \sum_v e^{-E(v,h)}
$$

$$
e^c v + bh + hWv \sum_v e^c v + bh + hWv = e^{(c + hW)v} \sum_v e^{(c + hW)v} = \text{sigm}(c_i + hW_i)
$$
In (eq 9) and (eq 10) the conditional probabilities are derived from Eq. 2 and Bayes’ theorem with the knowledge that \( v \) and \( h \) are independent. Most significantly, it is shown that the network distribution can be approximated by multiplying two sigmoid equations that are easy to compute.

\[
p(h_j = 1|v) = \frac{p(h_j = 1, v)}{p(v = 1)} = \frac{e^{-E(v,h)}}{\sum_h e^{-E(v,h)}} = \frac{e^{-E(v,h)}}{\sum_h e^{-E(v,h)}}
\]

\[
= \frac{e^{c'v+b+h+Wv}}{\sum_h e^{c'v+b+h+Wv}} = \frac{e^{h(b'Wv)}}{\sum_h e^{h(b'Wv)}} = \text{sigm}(h_j + Wv)
\]

These conditional probabilities are used in computing the gradient, specifically in sampling from \( v \) and \( h \) to arrive at the gradient of the energy function. Gibbs’ sampling uses an iterative approach of sampling from the data distribution represented by \( v \) and the hidden distribution represented by \( h \). This process is iterated as shown on right in Figure 5 and the sampled distribution approaches the stable distribution of \( p(v, h) \).

The gradient \( \frac{\partial E}{\partial \theta} \) is derived by taking the derivative with respect to each parameter in \( E(v,h) \) shown in (eq 11). These gradients are used to update their respective parameter value through gradient descent.
If we recall the loss function and the gradient operations, we can update the parameters to the right hand side of (eq 12) which was derived in Eq. 6, Eq. 7, and Eq. 8.

\[
\frac{\partial E(v, h)}{\partial w_{i,j}} = v_i h_j, \quad \frac{\partial E(v, h)}{\partial c_j} = v_j, \quad \frac{\partial E(v, h)}{\partial b_i} = h_i
\]  

(11)

The first term of the gradient (eq 12) within the brackets represents a conditional probability based on a single data vector. The second term represents the model distribution based on all of the training data. Since the second term would be difficult to compute in reasonable time, we approximate it using Gibb’s sampling using a single data vector. This iterative process of sampling hidden and visible units yields the stability distribution of the model. If we let \( < x >^0 \) represent the expectation of some data distribution \( x \) and \( < x >^{\infty} \) represent the expectation of the model, the update equations for learning can be written as (eq 13) for the weight transform parameters, (eq 14) for the hidden bias term, and (eq 15) for the hidden bias term.

\[
\Delta w_{i,j} \propto \epsilon (< v_i h_j >^0 - < v_i h_j >^{\infty})
\]

(13)

\[
\Delta c_i \propto \epsilon (< v_i >^0 - < v_i >^{\infty})
\]

(14)
A significant development in the use and performance of restricted Boltzmann machines was developed by Geoff Hinton (2) called contrastive divergence (Eq. 16). Instead of waiting for Gibbs’ sampling to converge to the stability distribution, the model parameters after k-steps of sampling are used as an approximation to the negative log-likelihood which has been shown to be convergent to the true solution (13). Contrastive divergence is important because not only is it convergent on the gradient but it greatly decreases run time. For these reasons, it is universally used in practice and sampling with k=1 seems to work well. In this case, the negative phase can be calculated using only four samples.

\[ \Delta b_j \propto \epsilon ( < h_j ^0 > - < h_j >^\omega ) \]  

(15)

\[ \Delta w_{l,j} \propto \epsilon ( < v_l h_j >^0 - < v_l h_j >^k ) \]  

(16)

\[ \Delta w_{l,j} \propto \epsilon ( E[ v_l^0 * P(h_j^0|v_l^0) ] - E[ P(h_j^k|v_l^0) * P(h_j^k|v_l^k) ] ) \]  

(17)

The equation (eq 17) is more of a trivial representation of the notation and clearly shows the connection to Gibbs’ sampling in developing the gradient. The power behind these machines resides in using the hidden representation of data as input to subsequent machines. Layering RBMs allows more abstract features to be developed, and Hinton has provided a practical guide for training these machines (14).

### 2.2 Convolutional Restrictive Boltzmann Machines

Convolutional RBMs (CRBMs) are similar to RBMs described in the previous section, except the weights are applied using a convolution operator and are shared across the image. CRBMs were originally proposed by Honglak Lee (9) and the description presented in this section is a summary based on his work (15).
CRBM uses a set of $K$ weight matrices of $N_w \times N_w$ weights along with a visible bias and $K$ hidden biases, one for each weight matrix. Each hidden unit is developed by a nonlinear convolution operation on the visible image and the visible image is reconstructed by a summation of convolution operations acting on the hidden units. The inverse transformation for binary input is also nonlinearly transformed to the $[0, 1]$ range while real valued input inverse is a purely a linear combination of features represented by the weight matrices.

An image of size $N_v \times N_v$ will produce $K$ hidden units of size $N_h \times N_h$ where $N_h = N_v - N_w + 1$. Since the hidden units are smaller than the visible units, the border is padded with zeros. The energy function of this network is defined similarly to the RBM, yet with a convolutional aspect involving the weights. $\tilde{W}$ is $W$ rotated 180 degrees or flipped vertically and horizontally. This phenomenon is due to the definition of convolution and its inverse over an image.

$$E(v, h) = -\sum_{k=1}^{K} b_k \sum_{i,j} h_{ij}^k - c \sum_{i,j} v_{ij} - \sum_{k=1}^{K} h^k \cdot (\tilde{W}^k \ast v)$$

Notice the similarity between the energy equation (eq 1) for the RBM and the CRBM energy equation (eq 18). There are still three terms to the parameter set, $\theta = (b, c, W)$ but $b$ is now a vector of size $K$ and $W$ is a set of $K$ weight matrices. The goal is still the same – to find parameters that minimize this energy function. Inevitably, optimizing this function will find weight templates that have a strong correlation to regions of images $v$; in this way image features are formed from the data in an unsupervised fashion.

These features are stored in the hidden units in a discrete binary format through the weights acting as correlation filters. Storing the hidden representation and the weights allows us to reconstruct the original...
image by summing the templates as directed by the hidden information. Having a common set of
templates used for all images permits a great reduction in the amount of information needed to store.

Gibbs’s sampling is used to compute the model distribution using conditional distributions derived from
the energy function and distributions. The derivation is essentially the same as the one for RBMs and
produces conditional probability equations Eq. 19 and Eq. 20. Note that when reconstructing the data
vector from the hidden representation, the hidden representation is padded with zeros of \( N_w - 1 \) width
on all sides in order to maintain the size of the original data from smaller hidden representation.

\[
p(v_{ij} = 1 | h) = \text{sigm} \left( \left( \sum_k W^k \ast h^k \right)_{ij} + c \right)
\]

\[
p(h_{ij}^k = 1 | v) = \text{sigm} \left( \left( \hat{W}^k \ast v \right)_{ij} + b^k \right)
\]

A hidden filter of size \( N_h \times N_h \) is produced for each of \( K \) features. This means that there exist \( N_h \times N_h \times K \)
hidden nodes to represent \( (N_h + N_w - 1) \times (N_h + N_w - 1) \) visible nodes. Since the number of hidden
nodes are approximately \( K \) times greater number than the number of visible nodes, the model is
considered over-complete. In order to learn generalizations quickly while minimizing repeated features,
the model is constrained so that only a sparse number of hidden units may be active. This is accomplished
by adding a bias term (eq 21) that reduces the chances of an activation potential triggering a hidden unit
by some user selected probability value \( p \). This probability term is a sparsity quantifier that enforces the
average activation to be close to \( p \) and thus limits the number of units that will be activated.
The training algorithm for a CRBM can be described in equations 22-24. These equations are similar to equation 16 for the restricted Boltzmann machine. $V^{(0)}$ is the current image within the training set. First, we compute the posterior probability $Q^{(0)} = P(H|V^{(0)})$ and sample from $Q^{(0)}$ to produce the binary $H^{(0)}$. $V^{(n)}$ is similarly sampled from $P(V|H^{n-1})$ and $Q^{(n)} = P(H|V^{(n)})$ and sampled to produce $H^{(n)}$. This is completed in an iterative fashion until $n$ is reached. Typically $n=1$ for contrastive divergence. This is Gibbs’ sampling from the RBM. Parameter learning for the CRBM is summarized in equations 22-24 which show how the contrastive divergence technique updates the parameters. Note that equation 22 is a weights vector of size $K$, equation 23 is a bias for each hidden term and equation 24 is a scalar used for reconstruction.

\[ \Delta b_k^{\text{sparsity}} \propto p - \frac{1}{N_H} \sum_{ij} P(h_{ij}^k = 1 | \nu) \]  \hspace{1cm} (21) 

\[ \Delta W^k \propto \frac{1}{N_H} \left( Q^{(0),k} \ast V^{(0)} - Q^{(n),k} \ast V^{(n)} \right) \]  \hspace{1cm} (22) 

\[ \Delta b_k \propto \frac{1}{N_H} \sum_{ij} \left( Q_{ij}^{(0),k} - Q_{ij}^{(n),k} \right) + \Delta b_k^{\text{sparsity}} \]  \hspace{1cm} (23) 

\[ \Delta c \propto \frac{1}{N_V} \sum_{ij} \left( V_{ij}^{(0)} - V_{ij}^{(n)} \right) \]  \hspace{1cm} (24)
These updates are applied to the parameters after summing gradients over a batch of images. This helps improve the accuracy of the weight adjustment and also helps limit the number of steps needed to converge to a solution. It also allows the use of a higher learning rate to accelerate the training process. After training a CRBM, additional machines can be stacked together to develop more interesting features. When these machines are stacked together it is often called a deep belief network.

3. Evolutionary Learning for Training Deep Networks
This thesis explores an evolutionary learning algorithm for training deep networks. The minimum developed by contrastive divergence as described in the previous sections is a gradient driven, direct descent to a minimum. Usually this minimum is close to the optimum especially when a decay term is present in the learning process. A decay term allows previous minima to be forgotten in exchange for lower minima solutions. A decay term allows the network to effectively “forget” the convergence in progress while moving toward better ones. This allows for a smooth descent to a low point in the parameter space. Momentum is also used to maintain a running average of the gradient which allows for a larger learning rate and a smoother descent to a minimum. Behind all of this, learning is driven by a single concept, summing a gradient over a batch of images.

This thesis explores alternative learning algorithms to contrastive divergence and is a continuation of our previous work using CRBMs on the SWAG dataset (16). The effectiveness of CRBMs suggest a number of interesting questions. Since contrastive divergence is in essence gradient descent, is it finding the optimum solution or just a local minimum? Can we accelerate the search for an optimum solution? Can we overcome the dependence on large data set? The goal is to see if evolutionary strategies could either learn better solutions or learn them in a faster manner than gradient descent. The breadth-first nature of
evolutionary algorithms may be able to better search the space for lower minima than gradient descent can do alone.

The goal of this thesis was to empirically evaluate evolutionary learning as an alternative technique to train deep networks for image processing. The desired outcome is to find more accurate solutions than the gradient descent based techniques using less computational resources and training data. The hypothesis is that the breadth-first search utilized by evolutionary algorithms may be able to more quickly explore search the space for lower minima than gradient descent can do alone.

3.1 Background on Evolution Learning
Evolutionary algorithms are an optimization technique inspired by biological evolution that uses an abstract representation of a solution to define the problem domain (17). The representation consists primarily of a set of mutable parameters that are gradually changed over time and evaluated by the optimization function. The problem space is represented by a set of parameters called genes. A set of genes makes up an individual which represents a single solution to the optimization problem. Using a set of individuals called a population allows many regions of the search space be evaluated at the same time. The genes of these individuals are gradually mutated or “evolved” to produce children where an individual is copied and then some of the copy’s genes are disturbed slightly creating a different individual. The optimization criterion or fitness function evaluates the worth of the new individuals along with the parents. Individuals that have fitness that is closer to a local optimum, have a high chance of surviving in the population. Some individuals in each generation are removed from the population and not allowed to reproduce.

As this iterative process takes place, gradually the fitness of the individuals increase as the population explores different regions of the search space. Diversity is maintained in the population in order to
adequately explore the search space, but as more individuals fall into some particular region of the search space (local optimum) the population slowly converges.

The major components of an evolutionary algorithm include: problem representation, a population of candidate solutions, operators to reproduce with variation and operator to select winners and losers. Picking a representation to define the search space is often the most critical step to develop an evolutionary learning algorithm. The representation must facilitate the evolutionary search process so small changes in an individual solution translate into small changes in the quality of the solution. The population is composed of partial or candidate solutions. They represent sample points in the search space defined by the representation. The variation operators typically include recombination and mutation. These operators are used to create new members of the population by either combining two or more existing solutions (recombination) or by varying an existing solution (mutation). Selection operators pick solutions for reproduction and/or pick members of the population to discard after reproduction essentially defining winners and losers. These ideas are explored in more detail below.

The choice of representation is problem specific. One of common approaches called Evolutionary Strategies (18) which use a representation based on real numbers. These real numbers compose an individual in the search space. The values of a single individual are referred to as a genotype. The individual real numbers called genes may be structured into higher order units called chromosomes, but however they’re structured, they represent something in the real world; the thing they represent is called the phenotype.

These values go through mutation and recombination processes in order to induce change in the population, but first it must be decided which individual to change; this is called parental selection. Although parental selection can be performed randomly, it can be a bias selection based on how well the parent represents the solution. This can be accomplished in a variety of ways. After parents are chosen, a
variety of mutation operations can be performed. Usually, changing a few genes by some variable amount is the most basic mutation. Depending on the structure of the genome, whole genes may be inserted or removed from chromosomes. Crossover is a recombination of chromosomes where some chromosomes are chosen from two or more parents to create a child. Mutation and crossover techniques are combined in a set ratio for producing children. The number of mutations to occur is determined by a geometric distribution (eq 25) where \( m \) is the number of mutations to occur, \( p \) is some probability determined by the one-fifth rule and \( X \) is a random variable.

\[
p(X = m) = (1 - p)^{m-1} p
\]  

(25)

Producing a large number of offspring helps maintaining population diversity. Also not producing enough children reduces the chances of moving in a better direction. In our system, many children are removed to prevent premature convergence and to prevent nonviable offspring.

After children are produced, there is the matter of deciding who survives for the next population. Survivor selection is usually performed on either the children alone or a combination of the children and parents based on their fitness score coupled with a stochastic algorithm that provide an opportunity for less fit individuals to survive. The goal of survivor selection is to create a population that is generally more fit than the previous population while maintaining diversity within the search space in order to prevent falling into local minimums.

4. System Description and Experimental Design

4.1 System Description

Two learning models are developed for comparison. First, a convolutional RBM is developed using the work of Honglak Lee (15) described in section 2.2. Second, an evolutionary system is developed that uses
the same weight structure but learns these weights through an evolutionary learning algorithm. Various experiments are performed using these systems in order to investigate the relative utility of the evolutionary design for learning image feature templates.

The differences between the two approaches to these models lie only in the learning mechanism. The system is composed of a set of \( K \) weight matrices, each being of size \( N_x \times N_y \) of real numbers, \( K \) hidden bias terms, one for each weight matrix and a visible bias term. There are two conditional probability terms that describe the transformation between the visible images and the hidden units. This remains the same between the two models. These conditional probabilities in (eq 26) and (eq 27) are derived from the CRBM model and used in the evolutionary alternative model to compute hidden units and reconstruction.

\[
p(v_{ij} = 1|h) = \text{sigmoid} \left( \sum_k W^k * h^k_{ij} + c \right) \tag{26}
\]

\[
p(h^i_{ij} = 1|v) = \text{sigmoid} \left( \sum_k W^k * v_{ij} + b_k \right) \tag{27}
\]

The process of learning the parameters \( W, c, b \) is the significant difference between the two systems. The sparsity term used to bias the hidden units in learning a unique solution is included in both systems.

However, in the CRBM system it is used in addition to a gradient term to determine the hidden whereas in the evolutionary system it determines the hidden bias (eq 28). The other parameters are evolved to align with the sparsity term. Enforcing sparsity is an essential need in this system and thus other evolving parameters are influenced by the hidden bias.
The sparsity term is important in developing unique features that can generalize across many samples of the data. The value of the sparsity constraint is chosen to prevent an identity function from being formed in the transformation parameters. The sparsity constraint works by indirectly effecting how many hidden units will become active on average. In order for it to be useful to develop unique features, the number of hidden units active must be significantly less than the average number of active regions in the data so that the weights will generalize to regions of the data and not just reflect a single value. An identity-like transform shown in Figure 6 is the most likely solution when equation 29 is true. The identity function will project the data onto the hidden unit in a one-to-one correspondence and can reconstruct using only one hidden unit. We prevent this by limiting the number of hidden nodes that may react to the image.

\[
\Delta h_k \propto \Delta h_k^{\text{sparsity}} \approx p - \frac{1}{N_h} \sum_j P(h_j^k = 1|v)
\]  

(28)

Several parameters influence the performance of the system. The most significant parameters are the sparsity constraint and the weighting of the terms used within the fitness function (eq 45). The sparsity constraint determines how often a feature can be found within the data. If the sparsity constraint is too high, possibly no features will be found present in the data and the reconstruction will be blank. Ideally,
features should match larger portions of the data, reducing the number of hidden units active. An example of the desired behavior is shown in Figure 7. The three leftmost images on the top row represent good, unique transform weights, that when convolved over the data produce the three images in the corresponding position in the second row. After hidden units are produced, you can see the three hidden values overlapped and color coded (bottom right image). This type of solution is preferred, such that the transform weights can identify spatial information in the data and hidden units can tell the presence of various features. This type of solution can only occur when the numbers of hidden units are constrained to be significantly less than the number of data items (eq 30).

\[
\frac{1}{N_n} \ll p \approx \text{Hidden}_{avg} \ll \frac{\text{Data}_{avg}}{K}
\]  

(30)

**Figure 7** Weights (top left), Expectation (bottom left), Faint Reconstruction (top right), Composition of Hidden Units (bottom right)

This can be seen even more clearly in the example of an handwritten eight (Figure 8). The template feature (second image from the right) is highly correlated to specific regions of the handwritten eight image (leftmost image). Even though other regions may respond to sections of the template, a threshold function is applied so only the regions showing a maximum response become active in the hidden layer
These regions are used to indicate where in the image the filter should be used in reconstructing the handwritten eight image. The sparsity constraint is what determines how much of the activated region falls above or below the threshold. Since we want a single filter to match only small region of the image, specifically the rounded edge, the sparsity parameter is set to a very small percent of the image. The reconstruction using this feature results in the image shown to the far right.

**Figure 8 The Original Image, One Feature Used, Hidden Activated Regions, and Reconstruction**

### 4.2 CRBM System Design

The convolutional restricted Boltzmann machine is trained by using contrastive divergence over a batch of images as described in section 2.2. These gradient estimate values are used to update the parameters by using a momentum term along with a weight decay term. Momentum allows previous gradients to be averaged into the current gradient preventing sudden shifts in the gradients and providing for a smoother trajectory across the weight space. While momentum may slow down learning, it will prevent bad gradients from having much effect and increases the likelihood of reaching a lower minimum by effectively not being able to slow down while passing local minima. Momentum is represented below by the $\mu$ term and it set to 0.5 in our system. The weight decay term works with momentum to forget a previous solution by slightly backing away from the current solution represented by the term $\xi$. For each gradient term, $\Delta p$ for parameter $P$, the following equations (Eq. 31, 32, 33) are used to update the target parameter.

\[
\Delta p = \Delta p \ast (1 - \mu) + \mu \ast (\Delta p_{old} - \xi \ast P)
\]  

(31)
The learning rate term is the step size in the direction of the gradient which is used to update the parameters. The smaller the learning rate, the longer the learning process will take and a learning rate that is too large will cause the system to become unstable. Using all of these terms, we can maintain a fairly large learning rate, especially near the beginning of training. As training progresses, the gradient updates are required to be more carefully applied as the system approaches a minimum, so the learning rate is gradually decreased to smaller, finely-tuned updates (eq 34). In this way, a layer can be trained within a few hours even on a single processor. In our CRBM we initialize the learning rate to a large value and slowly decrease it to Hinton’s recommended value (14) when the reconstruction error starts to increase. Since we expect the reconstruction error to decrease, we know the learning rate is too high when it increases. The learning rate is initialized to 10 at the start of training and eps is a differential value around 0.005.

\[ rate_{learning} = (1 - \text{eps}) \times rate_{learning} + \text{eps} \times |\hat{W}|/1000 \]  

The other terms found in the learning process are the momentum and the decay rate. These are set to 0.5 and 0.0001 respectively. The momentum term is increased to 0.9 when the change in reconstruction error on successive iterations falls below 0.0001 and was chosen based on Hinton’s work (14). The decay rate term allows the network to move away from premature solutions. This term is decreased as learning progresses alongside the learning rate (eq 35). As the system eventually begins to converge on an optimal solution, moving away from the solution with a decay rate becomes less necessary.
4.3 Evolutionary System Design
An evolutionary learning algorithm is also used to train our weights. The above weight matrices are represented as a summation of Gaussians in the genetic algorithm. Each Gaussian is treated as a gene in the genome for an individual of our population. The gene is mutated and corresponds to parameters of the Gaussian. The location of the Gaussian means within the weight matrix has been found to be the most relevant parameter to mutate. Each Gaussian is determined to be on or off based on a binary variable in a grid of size $K \times N_w \times N_w$ (Fig. 9) and their values are determined by an amplitude parameter.

These genes are used to determine the weight matrices (eq 36) where $k$ is the location of the gene (filter) within the genome and $A$ is the amplitude of a two-dimensional Gaussian.

\[
rate_{\text{decay}} = (1 - \text{eps}) \times rate_{\text{decay}} + \text{eps} \times rate_{\text{decay}}/2
\] (35)

\[
W^k = \sum_{x}^{N_w} \sum_{y}^{N_w} G_{xy}^k (A, 1) \times B_{xy}^k
\] (36)
There are $K$, 2-dimensional matrices each representing a filter in the CRBM. The binary matrix is used to simplify the representation of the genome to facilitate mutation. An amplitude value is used along with the binary matrix to form the real-valued filters used in convolution by summing Gaussians of height amplitude and positions binary. Other values that are mutated over generations are the visible bias, the threshold value for determining hidden units, and the amplitude itself. The hidden bias terms are determined by the sparsity constraint. They are always moved such that a user defined limited number hidden units are active at any given time. This constraint on the hidden bias is vital in producing unique features that match regions of the input data. Other constraints are imposed upon the system to support the learning process. For instance, hidden units are discouraged from being active in the same regions among the hidden layers preventing different layers from learning the same feature. Similarly, to encourage unique features to be learned, regions found in only one hidden layer are encouraged to activate. The goal is not only learn to reconstruct images, but to produce unique features that perform the reconstruction. The combination of these constraints prevents the system from learning an identity function (e.g. delta-like function).
A population of one hundred individuals is used in the evolution process as shown in Figure 10 in order to maintain a diverse population. Offspring are produced with a slight bias of picking parents with higher fitness before performing mutations and crossover. The number of offspring is a multiple of the size of the population, which is generally set to seven. Seven was chosen through experimental trials, mainly on the basis of reducing running time yet maintaining a number of children greater than the population size.

There are three types of mutations: adding new genes, deleting genes, and wide area mutation using two members of the population. Crossover of a single feature also occurs. The number of mutations is chosen using a geometric distribution with probability determined by the one-fifth rule. The one-fifth rule guides the number of mutations based on how quickly the system is learning. If the most-fit individual is surpassed by traditionally more than one fifth of its children, the mutation rate is decreased, else it is increased to maintain one-fifth of the children. Bits in the genome are turned on and off, either one at time or regions based on a probability map.
We defined an “edge based” probability distribution that is used to help determine which bits (Gaussian) in the gene to turn on and off for point mutation. This permits some structure in determining which pixels may be best to change. To encourage the evolution of contiguous regions of Gaussians within a template, we mutate bit patterns that define edge-like series of bits at a higher probability using a gradient approximation known as a difference of Gaussians (DoG) (19) and then scaling the result to form a probability distribution. The binary weights are included in the mutation equations to select weights that need to be flipped instead of a weight that is already in the opposite position—that is to say bits in the inactive position are the only ones to be flipped on and bits in the active position are the only ones to be flipped off. In (eq 37) only bits that are off have the opportunity to be turned on while in (eq 38) only bits that are on have the opportunity to be turned off.

\[\text{1. } \text{Flip On} = \max_{x,y,k} \left( -2 \ast W_{xy}^k + P \left( \frac{d^2W_{xy}^k}{dn^2} \right) + U(0,1) \right) \]  
\[\text{2. } \text{Flip Off} = \max_{x,y,k} \left( 2 \ast W_{xy}^k - P \left( \frac{d^2W_{xy}^k}{dn^2} \right) + U(0,1) \right) \]

These operations (Eq. 37, 38) are illustrated below in Figure 11. Adjacent bits have a higher probability of changing state while black space and white space have a lower probability of changing. These mutations allow continuous regions to form which have been observed to be more useful features. The effect of this operation is a single bit change allows gradual movement toward better solutions. These two mutations occur approximately one-third of the time.
In addition, there are two similar, region-based mutations used to thin or reconfigure the shape of the structure in the weights matrix or find common features between two feature sets. These two operations act on the entire weight space and not just a single bit. The first operation (eq 39) reconfigures the structure of a weight space by smoothing and resampling against a smoothed randomized space where $R$ represents the random space.

1. $B^k = G(B^k, 1) > G(R^k, 1)$  

2. $B^k = Dilation(B^k) \land Dilation(B^R_k)$  

The second operation (eq 40) is an “and” operation between two individuals of the population where $B^R$ represents a randomly chosen individual from the population. The “and” operation allows common features to be retained while removing extraneous features. Dilation of the binary weights expands the region of active bits in order to increase the amount of common material between the two individuals. Both of these operations can be seen in Figure 12. On the left, several bits flipped from the operation in equation 39 and on the right, active regions in common between two individuals are kept on while...
regions not held in common are turned off. Because of the larger effect of these mutations on the features space, it is uncommon for them to occur at around five percent of mutations.

**Figure 12 Gaussian and Dilation Mutations**

Lastly, two members are chosen for the crossover operation. A crossover takes a subset of filters from one individual and inserts them in the child (eq 41). In this case, only one feature is exchanged in crossover.

\[ B = B_0 \times B_R \times B^j K \]  

(41)

The crossover operation is displayed in Figure 13. Two members of the population are chosen, the one being mutated and one chosen at random; one of the filters are taken from the random individual and placed into the mutated member. This operation allows different features to work together to produce an effective solution. It is likely that good features are paired with not so good features and the crossover operation can relieve one from the other.

**Figure 13 Crossover Diagram**
The children produced from mutation operations are grouped with the parents in the new population and ordered by their respective fitness values. Progressive statistics are kept on correlation measurements between every member’s weight space. These statistics are used to remove less fit individuals that are too similar to other individuals. Generally, if an individual falls two standard deviations above the mean correlation to another individual, the less fit individual is removed. If the size of the pool of individuals falls below the population size, new individuals are added to increase genetic diversity. Less fit individuals are also removed if the weight genome is approaching an identity distribution or too many Gaussians have been added to the weight space. The removal is managed to prevent the population from losing too much of the previous generation yet to maintain genetic diversity. It is also managed to remove erroneous data and preventing the solution from zeroing out the weights. This process of creating a new generation can be seen in Figure 14.

![Figure 14 Process of New Generations](image)

The fitness function is a learning force in an evolutionary algorithm and is based on a combination of reconstruction error, sparsity, and properties of the hidden units such as overlap between features and
sparsity. The hidden layers are constrained to encourage the development of unique features. This is accomplished by preventing features from turning on in the same location within the hidden layer and encouraging features to turn on in unique locations within the hidden layer. The sparsity term influences the fitness function by favoring individuals closer to the target sparsity. Using sparseness limits the possibility of a trivial solution (e.g. delta function) and forces weights to diversify. However, the sparsity term alone is not enough to enforce a good solution and thus hidden constraints are also used.

The reconstruction error is arguably the most important term to reduce when it comes to training this system. Minimizing the reconstruction error is regularly done using a Euclidean metric between the two images. However, the MNIST dataset (Fig. 15) that is being used is composed of primarily black background causing even empty images to have a low reconstruction error. In order to remedy this, another error measure is constructed that emphasizes white space.

The error measure (eq 42) used treats background data so that it is just as important as foreground data by treating each separately as percentage of space. In addition, it minimizes smearing developed by reconstruction due to fuzziness and errors in the features. This measure has been shown to create perfect reconstruction by itself and is useful in progressive feature learning in our system. Let V represent a training image and \( \hat{V} \) represent the reconstructed image.

\[
\text{error} = \left[ 1 - \frac{\sum V \ast \hat{V}}{\sum V} \ast \frac{\sum (1 - V) \ast (1 - \hat{V})}{\sum (1 - V)} \right] + \frac{\sum (1 - V) \ast \hat{V}}{\sum (1 - V)}
\]  

(42)
This error is taken for each image in the training set and averaged before being used in the fitness function. Treating the white space to be as important as the black space allows for precision as a learning metric and prevents an all black reconstruction from receiving a high fitness value. An all black reconstruction allows weights to potentially go to zero and becomes a strong local minimum that is undesired. The multiplication used in equation 42 creates a maximum value when both foreground and background are matched.

There are two additional constraints included in the fitness function. The penalty term (eq 43) used to encourage uniqueness. This term penalizes features located in the same region of multiple hidden units (templates). The purpose of this term is to encourage weight templates to specialize to different regions of the data which implies that each weight filter is learning a unique feature within the data. This term is expressed as a difference between an overlap term and a uniqueness term.

\[
\text{constraint} 1 = \frac{2}{K^2} - \frac{1}{K} \sum_{m \neq n} \sum_{i,j} \frac{1}{N_{ji}} H_{ij}^m * H_{ij}^n \quad \text{min} \left( \frac{1}{N_{ji}} \sum_{i,j} \left[ \sum_{m} H_{ij}^m == 1 \right] pK \right)
\]  

The overlap term, seen as the positive term on the left of equation 43, is a measure of overlap between activated hidden units. When two layers are measuring a region in the image at the same time, the representation contains redundant information and is likely preventing other needed features from being represented. The ideal value for the overlap term is zero.
The uniqueness term, seen as the negative term on the right of equation 43, attempts to maximize activated regions of the hidden layers that are only found in one layer. A good solution will have several significant features that can be combined to reconstruct many different data samples. It is undesirable for one layer to be “on” and the rest of the layers to be off (zero). Therefore, a limit is placed on the value for the uniqueness term. The value is determined by taking the minimum between uniqueness and the sparsity measure times $K$. This caps the emphasis of the uniqueness term to the sparsity value for each of the $K$ layers effectively setting an upper bound for the sum of active unique units between the layers to $K$ times the amount of sparsity.

A second constraint (eq 44) is introduced to aid in feature specialization. This constraint encourages the system to use some percent of the hidden layers in reconstruction. For each training image, the average number of active layers is computed and a linear distance metric is computed from $\bar{H}$ where $\bar{H}$ is the average number of active hidden layers in the CRBM machine. This value was determined in running the CRBM machine in ideal conditions. By minimizing this constraint, the average number of hidden layers being used is encouraged to be the same as the CRBM model. This will help fewer features work together in forming an image and aid in preventing identity transforms in the weight space. Using all of the images in computing this metric allows some data to use less layers and others to use more. For instance, a “1”, may only need to use one feature while a “5” may have use four or five features for reconstruction.

\[
\text{constraint}^2 = 2 * \frac{\max(H, 1 - \bar{H})}{\max(\bar{H}, 1 - \bar{H})} + \frac{\max(H, 1 - \bar{H})}{p * \bar{H}}
\]  (44)

From experimentation, it is expected that two-thirds of the features will be used in reconstructing any given number on average. It is also expected that all of the features available be used on some digits.
Between these two restraints, only some features will be used for reconstructing a single data sample yet other features will be used to reconstruct other data samples. The filter used the least amount is expected to reach the target sparsity in approximately two-thirds of the data samples; that is any given filter is expected to be used two-thirds of the time. This combination will ensure that all of features are used but not all of the features will be used for every sample. It is desired for this value to approach zero, unlike constraint one, it is not normalized and gradually gets closer and closer to zero diminishing its value.

The fitness function (eq 45) consists of linear combination of several different measures. The ratios of these measures are important in training the system. While the reconstruction error is the primary objective to minimize, hidden overlap and entropy are important for developing unique and smooth feature spaces. The proper proportions, especially between error and overlap, are important to get right else the error will not be able to decrease further or some of the weights will not diversify. This proportion has been developed primarily by trial and error.

\[
\text{Fitness} = 100 - 50 \cdot \text{error} - 20 \cdot \text{sigma}(20000 \cdot \text{constraint1}) \\
- 30 \cdot \text{constraint2} - 10 \cdot |p - \text{sparsity}|
\]  

(45)

Ideally, all that would be needed is sufficient sparsity and the error metric and let evolution learn features that are forced to specify, but in reality they still have a tendency to develop delta-like functions and settle with an imperfect partial reconstruction before reaching a local optimum.

Being that these machines optimize on reconstruction error and properties of individual image, few images are needed to maximize the fitness function. The system will create features to mimic the precise data provided where the CRBM machine requires not only image gradients, but a large amount of gradients in order to find the correct direction that improves feature quality.
5. Experimental Results

Based on the two machines described, experiments are performed to compare their ability to reconstruct images as well as produce features useful for handwritten digit classification. Properties of evolutionary systems are focused on, especially population size, and number of children and their effect on the results.

First, reconstruction error is examined as a function of number of training images in the CRBM system along with the evolutionary system. Experiments were performed using twenty training images and then performed using a stream of imagery. Since the two systems work differently from each other, the CRBM is allowed to look at individual images and average over a batch before updating parameters while the evolutionary system is allowed to look at small batches of images for each iteration in order to find an accurate measure of reconstruction error.

Many comparisons can be made between gradient-based learning with a CRBM and our evolutionary based system. When only twenty images are used for training (Fig. 16), the evolutionary system performs significantly better than the CRBM system. The CRBM models the data distribution by maximizing the probability of data samples using a negative log gradient. In turn, the reconstruction error decreases because the model of the data is able to generate samples of the data. But in order for the CRBM to accurately reflect the gradient, it needs a summation of gradients from many, many images. Few images are not able to construct an accurate or reliable gradient for training the system. The evolutionary system only has to worry about the reconstruction of the data and is able to produce unique features by the constraints described above on the hidden layer. Since twenty images allow the system to construct an accurate measure of error, the system is able to improve through mutations.
A second population size is also displayed to show the relative improvement of reconstruction error between the two systems. A reconstruction error is able to reliably be determined using very few images while a reliable gradient term requires hundreds of unique images in order to develop an accurate gradient term. A larger population permits a broader sample of the search space and better reconstructive individuals are found with fewer steps. Many other factors could be improved to further increase the learning rate. Using the evolutionary strategy and only twenty images, the system reaches a minimum of 0.24 error. Just because the evolutionary system was effective at reducing the reconstruction error, there is no guarantee that this will result in useful features especially since only twenty images were used in the process. In fact, it seems that using so few images results in overfitting to those few images. Running test data through the trained system had a higher reconstruction error at 0.54.
Since the CRBM was not able to sample the distribution of the image space, it was not able to construct useful features (Fig 17), the feature it found most common in the small sample was too often overlaid into the weights causing them to converge to a single feature. The evolutionary system however was still able to diversify in order to satisfy the constraints enforced into the hidden layer of individual images. This allows the evolutionary system to construct a set of weights that are able to reconstruct the data. Note that the features developed are divergent from the CRBM features in many aspects, and not as elegant, but satisfactory for the experiment at hand.

For example, the evolutionary system was able to create these reconstruction images shown Figure 17 using the population shown in Figure 18. The population consisted of fifty individuals where each individual was composed of only five features.

The reconstructions shown above (Fig. 18) are obviously erroneous to a great degree, but could be greatly improved given larger populations and using more features. Most CRBM systems use around twenty
weight filters with the MNIST dataset. The population (Fig 19.) is not complete but is showing the diversity of the final population.

![Evolutionary System Final Population of 50 Members](image)

**FIGURE 19 EVOLUTIONARY SYSTEM FINAL POPULATION OF 50 MEMBERS**

Using larger amounts of data, the CRBM is able to develop a more accurate gradient for more efficient learning. Every feature available in the data is able to develop and no feature is overemphasized due to the large number of randomized images being presented. The error reaches a minimum of approximately 0.6. The CRBM is robust enough to jump over most local minima and settle at a near optimal solution but only when given an appropriate amount of data that can represent and thus generalize the training space. When enough data is provided, the CRBM strategy greatly outperforms the evolutionary approach (Fig 20) when it comes to features development and even reconstruction error.
The evolutionary technique is amended to use more data by averaging fitness scores over several batches of images through the training process. The large number of samples is able to accurately represent the image space and algorithm performance decreases in terms of reconstruction error. Having been presented with larger amounts of information, generalization of the image space is improved reducing fitness in exchange for potentially more robust features; thus allowing a momentum-like term in updating the fitness values has prevented overfitting the training data to a great degree. Since the evolutionary technique is driven by reconstruction error, overfitting consists of having an artificially lower reconstruction error due to few samples. The reconstructions themselves look similar to those shown in Figure 18 and a couple can be seen in Figure 22.

The weights developed by the evolutionary system are much more rugged looking, yet seem to perform a similar operation to their CRBM counterparts. In Figure 21, weights from each system are compared with their perspective twin. The gradient developed weights are smooth and clearly represent various regions
of the digit space while the evolutionary developed weights are jagged and discontinuous to a great degree.

*Figure 21 Weight Comparison – CRBM (top), Evolutionary (bottom)*

Sometimes it can be more difficult to perceive what pieces of the evolutionary weights are contributing to the feature. For instance, even two points can act as a feature and go unnoticed. While admitting neither as obvious nor consistent as the features developed by the CRBM, the developed system has found minimally constructed features to be acceptable. Certainly problems could arise with the much smaller correlation between the image and the weight matrix where the feature is concerned. In Figure 22, we can see that the diagonal of the seven has been identified by the two points shown on the right. The reconstructions suffer from this as shown on the left but the point is proven that evolutionary systems can develop at least minimally acceptable features.

*Figure 22 Constitution of a Feature*

The CRBM weights clearly show the power of gradient-based learning. While the CRBM reconstruction error does not fall much below the evolutionary system, the smoothness of the weights can prevent the reconstructions to not reach the maximum value of the original images. In Figure 23, CRBM
reconstructions are shown including the gradual fall off on the edges and partial reconstructions. The images displayed are real valued and the contrast between the number and the background may not be large enough to provide lower error rates with our custom reconstruction measure.

![Figure 23 CRBM Reconstructions of Digits](image)

**Figure 23 CRBM Reconstructions of Digits**

The oversight system that deletes individuals considered undesirable is overviewed in Figure 14. This system has played havoc with the worst fitness in the population because new individuals constantly migrating into the population with initialized random values as shown in Figure 24. The best fit value goes from around thirty-five (out of one hundred) to fifty-seven fitness over these iterations and then plateaus from there. It's hard to examine this because of the chaos from the lowest errors.

![Figure 24 Fitness Values per Iteration](image)

**Figure 24 Fitness Values per Iteration**
The evolution process goes through a series of states – from weight initiation to the final features used for reconstruction. In Figure 25 below, we show the population at three snapshots in the training process. The samples chosen to represent each segment of training were hand chosen from the population based on what is believed to be the best representation for the stage of learning.

![Figure 25: Samples through stages of evolution](image)

6. Conclusions
While evolutionary training has been shown to produce better reconstruction error using significantly less data, the system produces very different features than the gradient based system and may be prone to overfitting to small amounts of data. Training time does not seem to increase much as more images are used to calculate an average measure of fitness when keeping the same batch size. The system performs better when training on a fewer images but is prone to overfit the data. The complexity of tuning the
search and proportions within the fitness function is similar to the contrastive divergence counterpart. The system is useful when there is not enough training data for alternative methods.

Features for thinner digits have been found to be more difficult to produce than features for thicker digits. If only thicker digits are used in training, it is believed that more accurate results can be achieved. When both thin and thick digits are being used, a larger representation of the data set is required, especially for the thinner images. The two and the nine shown in Figure 26 provide an example of the contrast in the training data. It is difficult for the evolutionary system to accurately represent the two when most of the digits are thicker like the nine.

![Figure 26 Thick or Thin Images](image)

Using the first constraint, the system is able maintain features that correspond to unique regions of the image. However, the system is still able to duplicate features in different regions of the template. The following example (Fig. 27) creates a feature set that will be activated in two different regions of the hidden layer. When this happens, the first constraint is not able to aid in diversifying the weights and may hinder the development of good weights. This issue may have a greater impact in preventing good solutions until features can be “centered” in their respective weight space, or the first constraint is removed from the fitness function altogether. Controls such as the sparsity term limiting the number of active features and the desire to decrease reconstruction error are sufficient for training the system. It seems the first constraint may not be as important to learning as originally believed. Sparsity is enough to force the features to diversify.
General noise in the weight space may also be obstructive to the first constraint’s goal of diversifying the features. It is hard to tell what uniqueness in the hidden layer may imply when weights can be configured in so many different ways. While the principle is sound, it may be wise to devalue this constraint until it can be wielded in a more constructive manner.

One thought not included in the system was to base the probability distributions for mutation on the gradient terms. This would make a “guided” evolution toward a solution that moves in a stochastic-chosen subset of dimensions every iteration instead of using the entire gradient. This concept would require us to compute the entire gradient in order to do mutation and is more similar to the CRBM learning. It was believed to be unnecessary for training the population because the evolutionary process provides sufficient guidance in moving to an optimum solution with respect to the fitness function; in addition, it would detract from the originality of this thesis. This mutation process could begin with real-valued weight spaces that are updated with a partial gradient and even varying learning parameters. Random mutations could also occur in the form of adding and subtracting Gaussians. However, this was foregone by our evolutionary based implementation detailed here.

The concept of training CRBM models using evolutionary techniques is feasible and could save on training time. The main hurdle in further developing this type of system is innovating a fitness function that achieves all of the required properties of well-formed features yet accurately is able to generate data as well. While this paper has provided the basis of such a function, many improvements could be made in future research. Discovering how features are used in constructing an image is critical in developing these
constraints needed to guide the evolution process. Enforcing sparsity was shown to be critical in preventing identity transforms and forcing uniqueness among the features.

7. Future Works
This approach of using evolution in the training mechanism could be used in other machine learning applications and produce similar results. Recently, evolution strategies were applied to a RBM model (20) and will soon be applied to many other models in the future. The algorithm developed here could be improved in the future by reducing the complexity and constraints in the model thus allowing a more free evolution of features. Many of our mutations and our fitness function presumed a solution similar to what was already available which contained contiguous regions. We found that these assumptions may not be necessary to develop a good solution and features may be something harder to imagine than local regions of the data.
References


COPYRIGHT BY
RYAN MCCOPPIN
2014