Rochester Institute of Technology

RIT Scholar Works

Theses

2004

Concurrent evolution of feature extractors and modular artificial neural networks

Victor Hannak

Follow this and additional works at: <https://scholarworks.rit.edu/theses>

Recommended Citation

Hannak, Victor, "Concurrent evolution of feature extractors and modular artificial neural networks" (2004). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Concurrent Evolution of Feature Extractors and Modular Artificial Neural Networks

Victor Hannak

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
In
Computer Engineering

Approved by:

Principal Advisor Peter G. Anderson

Dr. Peter Anderson, Professor of Computer Science

Co-advisor Andreas Savakis

Dr. Andreas Savakis, Associate Professor and Dept. Head of Computer Engineering

Committee Member Shanchieh Jay Yang

Dr. Shanchieh Jay Yang, Assistant Professor of Computer Engineering

Department of Computer Engineering
College of Engineering
Rochester Institute of Technology
Rochester, New York
May 2004

RELEASE PERMISSION FORM

Rochester Institute of Technology

Concurrent Evolution of Feature Extractors and Modular Artificial Neural Networks

I, Victor Hannak, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Victor Hannak

Victor Hannak

5/19/04

Date: May 18, 2004

Abstract

Artificial Neural Networks (ANNs) are commonly used in both academia and industry as a solution to challenges in the pattern recognition domain. However, there are two problems that must be addressed before an ANN can be successfully applied to a given recognition task: ANN customization and data pre-processing.

First, ANNs require customization for each specific application. Although the underlying mathematics of ANNs is well understood, customization based on theoretical analysis is impractical because of the complex interrelationship between ANN behavior and the problem domain. On the other hand, an empirical approach to the task of customization can be successful with the selection of an appropriate test domain. However, this latter approach is computationally intensive, especially due to the many variables that can be adjusted within the system. Additionally, it is subject to the limitations of the selected search algorithm used to find the optimal solution.

Second, data pre-processing (feature extraction) is almost always necessary in order to organize and minimize the input data, thereby optimizing ANN performance. Not only is it difficult to know what and how many features to extract from the data, but it is also challenging to find the right balance between the computational requirements for the pre-processing algorithm versus the ANN itself. Furthermore, the task of developing an appropriate pre-processing algorithm usually requires expert knowledge of the problem domain, which may not always be available.

This paper contends that the concurrent evolution of ANNs and data pre-processors allows the design of highly accurate recognition networks without the need for expert knowledge in the application domain. To this end, a novel method for evolving customized ANNs with correlated feature extractors was designed and tested. This method involves the use of concurrent evolutionary processes (CEPs) as a mechanism to search the space of recognition networks. In a series of controlled experiments the CEP was applied to the digit recognition domain to show that the efficacy of this method is in-

line with results seen in other digit recognition research, but without the need for expert knowledge in image processing techniques for digit recognition.

Table of Contents

Abstract	3
Table of Contents	5
List of Figures	7
List of Tables	8
1. Introduction.....	9
1.1. Artificial Neural Networks	9
1.1.1. Artificial Neural Network Customization.....	9
1.1.2. Data Pre-processing / Feature Extraction	10
1.2. Evolutionary Processes for ANN Customization and Data Pre-processing	11
1.3. Thesis Overview	13
2. Background	14
2.1. ANN Customization.....	14
2.1.1. Topological Optimization	14
2.1.2. Parameter Optimization	17
2.1.3. ANN Encoding / Modular ANNs	18
2.1.4. Convolutional Neural Networks	19
2.1.5. Connectionist Theory Refinement	20
2.2. Feature Extraction	21
2.2.1. Dimensionality Reduction	22
2.2.2. Search Methods.....	22
2.3. Character Recognition	23
2.3.1. Image Preprocessing	24
2.3.2. Feature Extraction	25
2.3.3. Classification.....	25
2.3.4. Error Rejection.....	26
2.3.5. Comparative Performance Estimation	27
3. System Design	28
3.1. Artificial Neural Network	28
3.2. Data Pre-processing Network	29
3.3. Concurrent Evolutionary Process	31

3.3.1. Network Representation.....	33
3.3.2. Population Creation	34
3.3.3. Population Evaluation.....	35
3.3.4. Evolution.....	36
3.3.4.1. Parent Selection	36
3.3.4.2. Offspring Replacement Selection	37
3.3.4.3. Parent Crossover	37
3.3.4.4. Offspring Mutation	37
3.3.4.5. Offspring Evaluation.....	38
4. Empirical Analysis.....	39
4.1. Testing Methodology	39
4.1.1. Application.....	39
4.1.1.1. Database Partitioning.....	42
4.1.2. Fitness Evaluation.....	43
4.1.3. Testing Parameters.....	45
4.2. Testing Results and Analysis.....	46
4.2.1. Experiment #1 – Constrained CEP Execution.....	46
4.2.1.1. Baseline Test.....	47
4.2.1.2. Feature Extractor Contribution Analysis 1	48
4.2.1.3. Feature Extractor Contribution Analysis 2	50
4.2.1.4. ANN Contribution Analysis	53
4.2.2. Experiment #2 – Evolution of Optimal Recognition Networks.....	55
4.2.2.1. Evolutionary Process Analysis	55
4.2.2.2. Successful Recognition Network Properties Analysis.....	61
4.2.2.2.1. Recognition Network Computational Requirements	61
4.2.2.2.2. ANN Properties.....	62
4.2.2.2.3. Function Network Properties	62
4.2.2.2.4. Classification Error Analysis	63
5. Conclusions and Further Research.....	65
References.....	67
Glossary	75

List of Figures

Figure 1 – Fully Connected Artificial Neural Network	15
Figure 2 – An Arbitrary Recognition Network	32
Figure 3 – Flow Diagram of the Concurrent Evolutionary Process.....	32
Figure 4 – Sample Database Digits.....	40
Figure 5 - Handwritten Digit Database Example (of the number 2).....	41
Figure 6 - Baseline Test Success Rates.....	47
Figure 7 - Missing Feature Extractor Test vs. Baseline Test.....	49
Figure 8 – Missing Feature Extractor Test vs. Baseline Test	49
Figure 9 - Constrained Feature Extractor Test vs. Missing Feature Extractor Test	51
Figure 10 - Constrained Feature Extractor Test vs. Missing Feature Extractor Test	51
Figure 11 - Constrained Feature Extractor Test vs. Baseline Test	52
Figure 12 - Constrained Feature Extractor Test vs. Baseline Test	52
Figure 13 - Constrained ANN Test vs. Baseline Test.....	54
Figure 14 - Constrained ANN Test vs. Baseline Test.....	54
Figure 15 – Optimal Recognition Network Experiment Flow.....	56
Figure 16 – Evolution of Optimal Networks Test Unqualified Success Rates.....	58
Figure 17 – Evolution of Optimal Networks Test Qualified Success Rates.....	58
Figure 18 – Population Fitness at Each Stage of Evolution.....	59
Figure 19 – Rejected Digit Errors (A = Actual Digit, G = Best Guess)	63
Figure 20 – Accepted Digit Errors (A = Actual Digit, G = Best Guess)	64

List of Tables

Table 1 – Available Feuron Functions.....	31
Table 2 – Frequency of Digits in 15,000 Digit Database	40
Table 3 – Frequency of Digits in 2,000 Digit Subset of Database	40
Table 4 – Database Partitioning.....	42
Table 5 – Network Result Categories	44
Table 6 – Baseline CEP Test Parameters.....	47
Table 7 – Stage 1 CEP Parameters	57
Table 8 – Stage 2 CEP Test Parameters.....	57
Table 9 – Stage 3 CEP Test Parameters.....	57
Table 10 – Maximum Recognition Rates for Optimal Network Test.....	59
Table 11 – Time Requirements for Recognition Network Tasks	62

1. Introduction

1.1. Artificial Neural Networks

Artificial neural networks (ANNs) are mathematical models of the human brain composed of highly parallel processing elements with weighted interconnections. In recent years, ANNs have proven successful at solving a variety of classification problems due to their unique abilities. In particular, ANNs are effective at generalizing the input feature space, thereby enabling them to correctly classify noisy input vectors. Of particular importance is their ability to learn through programmatic modification of their internal connection weights based on feedback regarding a previous decision. Hence, ANNs can adapt to solve a variety of problems in a given domain without modification to their topology. This is especially useful in applications where the mapping function being approximated is dynamic. ANNs have been successfully applied to a range of real world problems from face recognition to controller design.

There are two major challenges that a system designer must overcome when implementing an ANN solution for a given problem. Although we consider each of these challenges individually, the system designer must address them simultaneously because their solutions are interdependent.

1.1.1. Artificial Neural Network Customization

The first challenge is the customization of the ANN for the target problem. There are two categories of ANN customization that need to be addressed: topological and behavioral. Topological customization involves selecting an appropriate architecture for the ANN. Traditional ANN terminology compels one to think of an ANN as a two-dimensional array of interconnected nodes such that the length of one dimension of the array corresponds to the number of layers, and the length of the second dimension

corresponds to the width of each layer. The number of layers, the width of each layer, and the density of the interconnections are typical topological parameters that need to be optimized in order to improve performance for a particular application. On the other hand, behavioral customizations refer to the optimization of parameters used by the ANN for learning and evaluation. Learning rate, learning momentum, and the specific evaluation function (sigma function) used by individual nodes throughout the system are examples of behavioral parameters that must be optimized. These customizations may be made globally, or each node may operate with unique (and possibly even dynamic) parameters.

One example of customization was an ANN designed by Le Cun et al. to recognize zip codes on hand-addressed envelopes [LeCun89]. This ANN topology had four layers containing 768, 192, 30 and 10 nodes respectively. However, not all the interconnections between nodes had variable weights. The first layer was actually connected to a 5x5 region of the input using 25 links with fixed interconnection weights. The purpose of this approach was to use the first layer of the network as a fixed set of feature detectors. This strategy proved successful with results of 99% accuracy with 12% rejection, but it is important to recognize the level of effort that was expended to customize this ANN specifically for its intended application.

1.1.2. Data Pre-processing / Feature Extraction

The second challenge in implementing an ANN solution for a given problem is the task of pre-processing that problem's data points for optimal application to the ANN. This process is called feature extraction. It is the process of rearranging or otherwise transforming each data point such that meaningful data can be extracted and irrelevant noise can be eliminated. The amount of pre-processing can range from simply transforming the input data into numerical input vectors, or it can completely identify the input data making the ANN unnecessary. Obviously, the latter scenario is usually unrealistic, which is why the ANN is required in the first place. Nevertheless, it

illustrates the wide range of possible solutions for the task of feature extraction. Additionally, it is important to consider that developing an appropriate pre-processing algorithm usually requires expert knowledge in a specific domain.

An example that demonstrates the importance of feature extraction is taken from previous research conducted by the author in the speech recognition domain [Hannak01]. An experiment was conducted where an ANN had to differentiate between sound samples taken from two individuals. Originally, the sound samples were introduced directly to the ANN in the time domain with no pre-processing. This yielded a recognition rate of 51%. However, by applying a Fourier transform to the speech samples prior to presentation to the ANN, the recognition rate improved to 90%+. Since the Fourier transform is a lossless transformation [Lathi92], it is clear that the presentation of the data, not just the data itself, is crucial to successful application of an ANN.

1.2. Evolutionary Processes for ANN Customization and Data Pre-processing

In an effort to find both the optimal ANN customization and the most appropriate data pre-processing algorithm for a specific application, two approaches can be used. The first is a theoretical approach where expert knowledge of the problem space can be combined with an experienced understanding of ANN behavior in order to generate an optimal ANN and data pre-processing algorithm. However, this approach is impractical for all but the simplest applications due to the complexity of the interaction between the ANN and the problem space. As a result, it is usually necessary to take an empirical approach whereby a search algorithm is employed to optimize at least a portion of the problem.

One such search methodology that may be applied to this task is an evolutionary process (EP). EP refers to a combination of programmatic operations, based on principles of natural evolution, used to search an unknown landscape. Genetic algorithms and genetic

programming are examples of EPs. It has been shown both theoretically and empirically that EPs are effective at parallelized search through a solution space for a global optimum [Goldberg89]. By starting with an initial random population and using search techniques such as crossover and mutation, EPs elegantly combine exploration with exploitation in order to avoid local maxima that would cause other search methods, especially hill climbing techniques, to fail. Additionally, due to their parallel nature, EPs are well suited for problems where little is known about the problem space, or where variables in this space are interdependent in unidentified ways.

In fact, the parallelism inherent in EPs makes them specifically suited for handling the problem of ANN customization and data pre-processing algorithm design. Oftentimes, the data pre-processing algorithm to be used for a specific ANN implementation is designed with little consideration given to how the algorithm will interact with the ANN. This is not intentional, but is due to the fact that the designer lacks the ability to forecast how changes to the algorithm would affect performance of the ANN under all the possible permutations of its parameters. Therefore, it is natural to try to automate the task of deriving a pre-processing algorithm so that these permutations can be quickly explored.

At this point, it makes sense to consider an example of successful evolution in nature to further validate this approach. The human visual system, consisting of the eyes, optic nerves and visual cortex, is such an example. The visual cortex and other brain areas that deal with image processing did not evolve independently of the eyes themselves [Bownds99]. Rather, these two organs evolved concurrently. Therefore, when designing a system based on the evolution of the human sensory organs, it makes sense to allow all of the components of the system to evolve interdependently. It is intuitive that any component that is fixed in the design process will become a limiting factor of the design. By minimizing the number of static components in the system, the designer can exploit the full parallelism provided by EPs in search of the optimal solution.

1.3. Thesis Overview

The intent of this thesis is to prove that concurrent evolution of ANNs and pre-processing algorithms can yield an effective recognition system without the need for expert knowledge about the problem domain.

In order to validate this theory, a concurrent evolutionary process (CEP), that can simultaneously evolve an ANN along with data pre-processing algorithms, is designed and described. This process is then applied to the task of handwritten character recognition. The CEP is tested under a variety of conditions in order to validate its effectiveness.

2. Background

Three major areas of research need to be examined relative to the content of this thesis. The first is ANN customization, which includes a broad range of sub-topics including evolution of ANN parameters and the use of modular ANNs. The second area is that of generic feature extraction. Although this thesis specifically focuses on digit recognition, the methods developed herein apply to any application domain, and it is important to consider other research in feature extraction that is application independent. Finally, it is also necessary to review other attempts at character recognition to examine the strengths and weaknesses of other approaches to the same challenge. The task of digit recognition is also specifically addressed to provide some perspective on system performance.

2.1. ANN Customization

Considerable research has been conducted on methods for customizing ANNs for a specific application. Several of these approaches will be examined and compared to the CEP suggested in this thesis.

2.1.1. Topological Optimization

There is an abundance of research on the optimization of ANN topology to target a specific application. Topological components subject to optimization include:

- Input nodes
- Hidden nodes
- Layers of hidden nodes (this is just an abstraction)
- Bias nodes
- Interconnections

Figure 1 shows a specific example of a neural network topology that has five input nodes, three output nodes, and two hidden layers of four nodes each. Each adjacent layer is fully interconnected and a bias node is not present.

Figure 1 – Fully Connected Artificial Neural Network

There are generally three approaches for searching the domain of ANN topologies: constructive methods, pruning methods and evolutionary algorithms.

Constructive algorithms begin with a minimal network topology and accomplish optimization through the addition of topological components. Examples of constructive methods include Dynamic Node Creation [Ash89], the Cascade Correlation Network [Fahlman91] and the Quasi-Newton method [Setiono95]. Conversely, pruning algorithms start with a large network and delete components to improve ANN performance. Examples include Sensitivity Calculations and Penalty Term Methods [Reed93]. One of the disadvantages of pruning algorithms is the computational requirement for training large networks in the early stages of pruning.

Another disadvantage that is shared by both constructive and pruning methods is due to the hill-climbing search mechanism employed by these methods to find the optimal

solution. The problem is that hill-climbing techniques are susceptible to becoming caught in local optima [Angeline94]. To illustrate this point, consider an example of a constructive method. It is possible that, for a simple application, an ANN with a single hidden node will perform better than most multiple hidden node solutions. However, if the real global optimum is a complex network with many nodes, it is unlikely that the search algorithm is capable of crossing the plateau between optima.

The third approach for searching the domain of ANN topologies, the use of evolutionary algorithms, attempts to address the shortcomings of the previous approaches. Given that each point in the search space represents some combination of network performance criteria (i.e. complexity, learning rate, success rate, etc), all the points form a surface. Miller et al. [Miller89] describe the characteristics of this surface that make evolutionary algorithms particularly suited to the task of ANN evolution. These characteristics are:

- The surface is infinitely large since the number of possible nodes and connections is unbounded.
- The surface is non-differentiable since changes in the number of nodes and connections are discrete and can have a discontinuous effect on the neural network's performance.
- The surface is complex and noisy since the mapping from an architecture to its performance is indirect, strongly epistatic, and dependent on the evaluation (training) method used.
- The surface is deceptive since similar architectures may have quite different performance.
- The surface is multi-modal since different architectures may have similar performance.

Because the search space is so large, enumerative methods are highly inefficient. Also, since the search space may not be differentiable, hill-climbing or gradient descent methods are likely to fail [Miller89]. Therefore, a search method that combines random search with hill-climbing techniques offers substantial advantages.

Evolutionary processes use a population of initial starting positions and the mechanism of crossover to search the surface. This technique limits the likelihood of becoming caught in a local optimum. A variety of research touts the benefits of EPs in ANN creation along with numerous suggestions for improvement [Angeline94][Yao97]. However, there is a distinct lack of direct comparison between EPs and other constructive and destructive methods probably due to the difficulty of recreating exact testing environments.

2.1.2. Parameter Optimization

Similar to topological optimization described earlier, the parameters that control ANN behavior must also be optimized to improve performance. Some of the parameters that can be adjusted are:

- Initial starting weights of interconnections
- Learning rate and momentum (used in adjusting interconnections during gradient descent back-propagation)
- Activation function of the nodes

The concurrent evolution of parameters and topology is addressed in a variety of research [Angeline94] [Maniezzo94] [Fogel95] and has been shown to present a number of benefits. For example, by simultaneously evolving weights and topology, it is possible to reduce the problem of noisy fitness evaluations due to the random distribution of weights that would otherwise be necessary [Yao97]. The concurrent evolution of node transfer functions and topology is similarly beneficial by creating ANNs with improved generalization [Liu96].

2.1.3. ANN Encoding / Modular ANNs

A variety of encoding schemes can be selected to encode ANNs for the purpose of evolution [Yao97]. These range from indirect encoding to direct encoding. Indirect encoding limits the representation to encode only certain critical information like the number of hidden units or the hidden layer depth. On the other hand, direct encoding specifies each component in the ANN and its corresponding parameters. The advantage of the latter approach leverages the ability of EPs to reproduce effective sub-strings or building blocks within the population [Balakrishnan95]. By using direct encoding, it is possible for the EP to create highly successful offspring by combining building blocks found throughout the population. In this way, modular networks can be evolved such that each module is effective at a specific sub-task. For example, in digit recognition, if parent A has a module that is effective at recognizing the digit two, and parent B has a module that can recognize the digit nine, then a crossover of parent A and B can yield offspring that is effective at recognizing both digits two and nine. Modular networks have been shown to yield good results [Darwen96] [Gruau95], especially when the topology is concurrently evolved with the parameters [Liu97].

On the other hand, a disadvantage of direct encoding is the permutation problem [Hancock92][Schaffer92][Belew91]. Simply stated, an indirect encoding may allow numerous representations of a specific ANN. Consequently, inefficiencies are introduced into the crossover operation, not to mention the inefficient creation and storage of diverse populations. Therefore, it is important to determine the encoding scheme intelligently in order to minimize these inefficiencies.

The concept of modular ANNs can be further extended into the domain of network ensembles. An ensemble is a combination of multiple networks in an attempt to improve system performance. At this juncture, it is important to point out the distinction between ensembles and modular networks. While each component in a modular ANN generally solves a unique sub-task, all the components in an ensemble generally solve the same task. The purpose of a module is to augment the capability of an ANN whereas the

purpose of a single ANN in an ensemble is to improve generalization. A variety of methods are available to accomplish the combination of multiple networks to create an ensemble [Sharkey96][Hashem93] including averaging, non-linear combination and stacked generalization. Additionally, it is important that the networks are diverse in order to maximize benefit [Sharkey97][Opitz96].

Although ensembles are generally treated as distinct from modularization in the research literature, it seems forthcoming that linear combination ensembles could be naturally generated in a modular approach. Additionally, given the availability of special combination functions, even non-linear combination would be possible. Although this approach is not addressed within the scope of this thesis, it does appear to have merit for further study.

2.1.4. Convolutional Neural Networks

Convolutional neural networks (CNNs) are another form of neural network that have similar properties to the recognition networks that are described in this thesis. This type of network was developed to overcome shift and deformation variances in the input data through the use of local receptive fields, shared weights, and spatial sub-sampling [Lawrence97]. CNNs are especially beneficial for applications where high dimensional input data is applied directly to the neural network without the prior step of pre-processing the data [Neubauer98]. Thus, the data pre-processing is embedded into the neural network by carefully customizing the network architecture.

CNNs use a layered architecture such that each layer alternates between a convolution function and a sub-sampling function. In this way, each convolutional layer is responsible for identifying progressively larger features in the input space composed of smaller features extracted by the previous layer. The final layer of neurons is a fully interconnected network that is used to classify the data according to the extracted features. By using fixed weights for the convolutional and sub-sampling layers, the

network will not merely learn localized features, but must extract more position invariant features thereby offering improved generalization.

There are some similarities between CNNs and the recognition networks that are described in this thesis. Firstly, when applying the data to a recognition network to evaluate its fitness, the data is applied circularly, thereby convolving the data with the data pre-processing network. However, there is only one layer where convolution occurs, whereas CNNs repeat the convolution process across several layers. Furthermore, the recognition networks described in this thesis do not have the capability to allow weight sharing between interconnections, although the data pre-processing networks do provide the ability to perform sub-sampling. Also recognition networks allow more complex mathematical functions to be used for the convolution, thereby creating a more compressed network that can extract more complex features from the input data. Finally, it should be noted that the process described for evolving recognition networks does not preclude the design of recognition networks with CNN properties like weight sharing, but the implementation used for this thesis does not include these capabilities.

2.1.5. Connectionist Theory Refinement

Finally, it is also necessary to consider connectionist theory refinement as an alternative or an improvement to the CEP approach. Connectionist theory refinement applies traditional theory refinement to the creation of ANNs. Traditional theory refinement uses a set of rules provided by expert examination of the problem domain to create an accurate predictor. Once the predictor has been applied to test data, the rules can be refined based on the system performance to improve the generalization ability of the system or to correct gross errors. Connectionist theory refinement uses the same approach with the exception that the predictor is constrained to be an ANN such that the rules are expressed through the network's topology and initial weights. It has been shown that ANNs that take into account expert knowledge of a problem domain not only perform more accurately than generic ANNs, but also converge more quickly [Towell90].

Improvements to this approach involve the use of EPs to search the domain of knowledge-based ANNs to select the most successful members, which may use only a subset of the available features [Opitz97] [Burns98].

It is inappropriate to directly compare knowledge-based methods with those that use empirical derivation, because the presence or absence of expert knowledge in a particular domain varies. If such expert knowledge exists, then it makes sense that it be used in some manner to improve system performance. On the other hand, the relationship between the domain and the ANN may be so complex that it is unclear how the expert knowledge may be applied. Or, even worse, there may be no expert knowledge about the domain at all. Therefore, it is appropriate to develop a method for customizing ANNs regardless of the presence of expert knowledge, so that such a system can be used in all applications. Then, if necessary, the system can be modified to integrate expert knowledge for a particular application. Therefore, it can be concluded that knowledge-based methods should be considered only as an improvement to CEP rather than an alternative.

2.2. Feature Extraction

There are a variety of methods and motivations for extracting relevant information from raw data for the purpose of classification. The process of feature extraction covers a broad range of techniques including the following:

- Selecting a feature subset from raw data
- Creating features through linear combinations of raw data
- Creating features through non-linear combinations of raw data

It is necessary to employ these pre-processing methods for a variety of reasons. For example, a recognition system may have constraints limiting time to compute a match, thereby limiting the number of features that can be processed in the available time. This

may require subset selection in order to minimize the number of features. Another possibility is the need to augment the feature set due to missing or invalid data. This is called feature construction [Liu98a] or function decomposition [Zupan98].

2.2.1. Dimensionality Reduction

One of the most frequently used techniques for feature extraction is dimensionality reduction [Scheunders98] [De Backer98]. Dimensionality reduction is the process of reducing the dimension of the feature space in an attempt to improve visualization of high-dimensional data sets. Although the easiest way to accomplish this task is through subset selection [Devijver82], numerous other linear and non-linear approaches are available as well, including principal component analysis, multidimensional scaling, Sammons mapping, self-organizing maps, or auto-associative feedforward networks [Scheunders98]. Each method has its advantages and disadvantages depending on the properties of the application such as dimensionality and quantity of data points.

2.2.2. Search Methods

In order to find the appropriate feature extraction method for a particular problem, it is necessary to search the solution space of feature extractors. Subset selection can be used as a trivial example to demonstrate this process [Langley94]. In this case, each point in the search space represents one possible subset of the raw data. A variety of search methods can be employed to find the optimal solution including random search or incremental techniques [Liu98b]. However, this search space has similar properties to the ANN customization search space and, therefore, evolutionary processes are again appropriate to conduct the search [Chang90].

For some applications, it may be necessary to move past the trivial subset selection strategy into more advanced linear and non-linear algorithms for feature extraction. At this point, the search space expands substantially to include not only the raw data, but also a variety of mathematical operators that act upon the data. Although this makes the search more difficult, it does not necessarily impart a need to change the search strategy. This is supported by empirical evidence that validates the effectiveness of using evolutionary processes for finding appropriate features [Firpi99] [Gaborski93] [Chang90].

2.3. Character Recognition

Artificial neural networks and feature extractors have been applied to a variety of problems in the pattern recognition domain [Russell95]. However, the problem of character recognition is ubiquitous in pattern recognition research literature because of its broad application and the abundant availability and straightforward classifiability of test data.

Historically, a variety of approaches have been tried to achieve accurate character recognition. Initial attempts were mostly template matching methods that were severely limited by available computational power and therefore employed only simple logic operations [Mori92]. However, these original attempts were only suitable for recognizing typewritten digits. As computing power improved the challenge of recognizing handwritten digits was broached with structural analysis techniques. These methods isolated unique structural characteristics of each character and used this knowledge to classify test data.

Today, a typical recognition system is loosely comprised of three stages: image preprocessing, feature extraction, and classification. This partitioning is rather arbitrary and oftentimes tradeoffs are made that change the distribution of computation between each stage. For example, a neural network can be considered to be simply a non-linear

feature extractor [Cherkassky98][LeCun89]. Nevertheless, for the purpose of organization, the aforementioned partitioning will be preserved for the remainder of this discussion.

Substantial research from the 1960's through today has yielded a variety of new methods for each stage of a character recognition system [Mori92]. The research relevant to each stage of the system is subsequently described, followed by a discussion of error rejection and comparative performance estimation.

2.3.1. Image Preprocessing

The first step of processing a character is to take the raw image data generated by a sensor and convert it into a form suitable for digital manipulation. But before a character can be processed further, it must be isolated from other data and noise surrounding it. This process is called segmentation. Segmentation is a daunting task and a full analysis of it is outside the scope of this thesis.

In the next step of image pre-processing, the image is normalized by sizing, centering, and rotating the character. Operations such as low pass filtering to reduce stroke widths may also be done during this step. Most of the algorithms used for normalization are commonplace image processing techniques such as Gaussian filtering to smooth the image and Laplacian transforms for edge detection [Breuel93]. However, some of the algorithms are more advanced and specifically targeted for character normalization. For example, extensive research has been done on the use of moments for normalization of characters [Alt62] [Casey70].

The output of the image pre-processor is usually a binary image formatted specifically for proper application to the feature extractor. In some cases, the output is not limited to binary data in order to allow for greater expressiveness of the features [Knerr92]. Note that the ordering of the image pre-processing tasks varies for each implementation.

2.3.2. Feature Extraction

A variety of feature extraction methods have been tried in an attempt to improve classification rates for character recognition tasks. Simpler approaches to this task involve selecting a subset of pixels from the pre-processed image or determining relationships between pixel locations to extract structural information [Behnke97] [Mori92].

However, these simpler techniques are susceptible to minor variations like offset or skew in the sample data. Consequently, more advanced techniques have been developed that are tolerant to these types of variations. For example, the use of Fourier descriptors [Lai81] can be used to extract approximations of the shapes that are present in the image regardless of placement. Additionally, moments have invariant properties that hold regardless of variations in size, translation and rotation that make them good candidates for feature extraction as well [Belkasim91] [Cash87].

2.3.3. Classification

The final step in character recognition is classification. The classifier is used to categorize the data based on the features extracted from the character. The simplest methods for doing this involve calculating the distance of the current character's feature values from the feature values for one or more ideal representations of valid characters. Then, the ideal character whose feature values are closest to those of the current character is selected as the best guess of the classifier. A variety of parametric methods are available to quantify distance between feature sets, including Euclidean distance, cross correlation, and Mahalanobis distance [Cash87].

Unfortunately, these parametric methods fall short in many real world applications because they assume unimodal density functions [Duda00]. Therefore, non-parametric techniques like nearest-neighbor classifiers are better for applications with arbitrary

distributions like character recognition. A good comparison between the nearest neighbor classifier and some parametric techniques can be found in [Grother93].

Another non-parametric technique that has received much attention in the character recognition research literature is the use of ANNs [Denker89] [LeCun90] [Knerr92]. Generally, ANNs are used for the role of classification, but some research indicates that they are also effective at processing low-level information, thereby bypassing the need for some feature-extraction methods [LeCun90]. The major criticism of ANNs is that the back-propagation algorithm that is most commonly used for learning is rather slow [Fahlman91].

Given that each classification approach has unique advantages and disadvantages, it is natural to try to combine multiple classifiers to improve recognition accuracy. A number of efforts have been made in this area, with experimental results showing that combined classifiers are more effective than the best individual classifier [Behnke98] [Vuurpijl98].

2.3.4. Error Rejection

Regardless of the type of classifier that is selected for the recognition task, it is important that the best guess of the classifier be accompanied by some confidence measure. Then, the best guess can be rejected if the confidence of the classifier is below a selected threshold, in order to improve the classifier's success rate. According to LeCun et al., "In a realistic application, the user is less interested in the raw error rate than in the number of rejections necessary to reach a given level of accuracy" [LeCun89]. This statement outlines the importance of understanding the error-reject tradeoff, but unfortunately, very little research has been done in this field. One paper that does address this issue, however, finds that the error-reject tradeoff of a variety of classifiers are strikingly similar [Hansen97].

2.3.5. Comparative Performance Estimation

Each time a new character recognition system is designed, it is essential to compare the system to others found in previous research. However, this task is a significant challenge for a variety of reasons. First and foremost is the issue of inconsistency between the databases used for testing of different recognition systems [Denker89]. It is possible to create a database that contains characters that have been carefully written such that any recognition system can offer excellent results. On the other hand, many databases taken from real world contain numerous examples of characters that are not even discernible by human readers. Other factors that may inhibit an accurate comparison include test set balance (i.e. how many of each number), geographical origin of the data, and independence of the training and testing sets [Suen92]. For these reasons, it is important to qualify testing results with the characteristics of the database used.

Nevertheless, some comparison is necessary to get a general feel for a system's relative performance. Most of the results available apply to the character recognition subtask of digit recognition. A good table comparing digit recognition results found in research is available in [Suen92]. Typically, the databases used for testing consist of segmented digits that are handwritten, typed, or a combination of both. Generally, most recognition rates are above 86% accuracy with rejection rates below 12%. Some research even claims recognition rates of 99.5% with no rejection [Duerr80]¹.

¹ The method proposed in [Duerr80] implements a multiple stage classifier such that correctly identified samples are not passed on to subsequent stages. In a real testing environment, there is no capacity to verify correctness prior to further processing, therefore, the reported accuracy appears to be artificially inflated compared to other results.

3. System Design

In order to test the assertions of this thesis, it was necessary to develop a software implementation to concurrently evolve ANNs and feature extractors. A flow diagram for this system is shown in Figure 3.

Each of the system components was developed as a C++ class. In total, there were 9 classes comprising 5227 lines of code. The GNU gcc compiler was used to compile the code and testing was done on a Linux operating system.

Each major component of the system is subsequently described. However, for the purpose of this thesis, the implementation details of the system will only be described insofar as they pertain to its performance and the conclusions reached as a result. Therefore, specific implementation details, such as how the code was partitioned, are outside the scope of this thesis. On the other hand, architectural details, such as the encoding mechanism used by the CEP to specify an ANN, are relevant to the performance of the system and therefore must be discussed.

3.1. Artificial Neural Network

In order to achieve an ANN that is highly configurable, the concept of a self-contained ANN node (neuron) was introduced. Each neuron contains an algorithm for evaluation, an algorithm for learning, parameters for execution and pointers to other neurons that are connected to its inputs and outputs. The parameters and connectivity pointers are configurable for each neuron, thereby making a generic node that could be modified to meet a variety of requirements. For example, each node can have a unique learning rate, momentum and sigma function. Also, since the input and output pointers are implemented as linked lists, there is no limitation (other than computing resources) with regard to layer depth, layer size, and layer interconnectivity. In fact, the ANNs created

by this system need not conform to the traditional multi-dimensional array layout that supports connections only between adjacent rows. Rather, these ANNs can be connected as any complex graph. It is therefore important to ensure that only acyclic graphs are created when interconnecting neurons in order to prevent feedback loops that would cause the normal feed-forward operation or back-propagation training methods to run infinitely. This challenge is addressed in the EP encoding section.

All inputs to the ANN must originate directly from the raw or pre-processed data values. These values are referred to as features because they represent some transformation of the raw data. The methods by which these features are created are addressed in the next section. Neurons that contain input connections to features are referred to as input neurons.

All outputs from the ANN must terminate at specific output neurons. There is exactly the same number of output neurons as there are possible classes in the sample data. In the case of digit recognition there are 10 output neurons, one corresponding to each possible digit.

The evaluation cycle of the ANN is a parallel flow of activations from the network inputs to its outputs. The training cycle of the ANN is a parallel flow of weight updates from the output neurons to the input neurons. The weights are updated throughout the network based on gradient descent back-propagation.

3.2. Data Pre-processing Network

The data pre-processing network was designed in a similar manner to the ANN. Individual function nodes (feurons) were created that have configurable function algorithms and pointers to other feurons that are connected to their inputs and outputs. Each feuron can therefore perform some simple mathematical operation on the raw input data. By combining feurons, more complex mathematical operations can be performed.

Of course, similar to the ANN portion of the system, it is necessary to ensure that only acyclic graphs are created.

Another important ability of feurons is their ability to collect data before triggering their output connections. This feature gives feurons the ability to perform mathematical operations like averaging. The number of data elements that each feuron collects is configurable.

Also, prior to operating on its inputs, a feuron will multiply each input by a configurable multiplication constant and add a configurable offset constant. This allows scaling of the feuron's inputs prior to any mathematical operation.

The function that each feuron performs is fixed when the network is created. Table 1 shows a list of available feuron functions. For comparison operators like greater than or less than, all input values to the feuron are summed and compared to 0. If the comparison yields true, a constant is output. If it yields false, the negative of the constant is output. The constant is also configurable along with the feuron function. There is no need to provide the ability to compare against values other than 0 since the inputs can be offset by positive or negative values prior to the comparison.

All inputs to the data pre-processing network must originate directly from the raw data values. Feurons with connections to the raw data are called input feurons. The manner in which the raw data is applied to input feurons is quite complex and merits further discussion since it relates to the performance of the system.

1. Pass through	13. Arc tangent
2. Average	14. Arc tangent y/x
3. Greater than	15. Arc tangent x/y
4. Less than	16. Hyperbolic sine
5. Absolute value	17. Hyperbolic cosine
6. Ceiling	18. Hyperbolic tangent
7. Floor	19. Power
8. Sine	20. Power inverse
9. Cosine	21. Exponent
10. Tangent	22. Log
11. Arc sine	23. Log base 10
12. Arc cosine	24. Square root

Table 1 – Available Feuron Functions

All outputs from the data pre-processing network must terminate at specialized output feurons (these are the extracted features). These feurons contain pointers to one or more input neurons in the ANN. The number of output feurons is completely configurable.

3.3. Concurrent Evolutionary Process

The task of the concurrent evolutionary process (CEP) is to search the domain of recognition networks in order to find networks that are effective at recognizing the application test data. A recognition network consists of a data pre-processing network connected to a modular ANN. An example of a single recognition network is shown in Figure 2. Also, Figure 3 shows a flow diagram for the CEP.

Figure 2 – An Arbitrary Recognition Network

Figure 3 – Flow Diagram of the Concurrent Evolutionary Process

Although it is possible to create and manipulate a recognition network directly for the purpose of evolution, this is computationally impractical when dealing with large numbers of networks. Therefore, the CEP works with an encoding of the recognition network rather than the network itself. This is an important distinction because it implies that the capability of the CEP is limited by the robustness of the encoding mechanism. Consider an example application where a ten-node recognition network is the optimal solution. If the encoding language has no way of expressing a network with more than eight nodes, then there is no possibility that the optimal network can be found.

It is also necessary to clarify the definition of concurrent evolution as it is used to evolve optimal recognition networks. It is important to note that each string that is manipulated by the CEP is a complete recognition network containing both neurons and feurons. Therefore, concurrent evolution is the simultaneous evolution of interacting components within a string. It is not co-evolution, whereby networks would evolve independently and interact only during fitness evaluation. In fact, individual recognition network strings or their corresponding network implementations never interact outside of the evolutionary crossover operation. More specific details on string representation are subsequently detailed.

3.3.1. Network Representation

In order to understand the operation of the CEP, it is first necessary to explain the encoding used to describe the recognition networks that it creates and evaluates. It was decided that the best method for network encoding would be through the use of a string structure that was specially designed to simplify many of the CEP tasks. This approach has several advantages due to the ease by which strings can be created, manipulated and stored. Each string represents an entire recognition network and must be encoded in a specific manner.

At a high level, each string represents one or more paths from an output neuron to a raw data location. Therefore, no runt paths can exist. The paths may share nodes (neurons or feurons), but they cannot be cyclic. In the case that a path repeats, the connections are not duplicated.

The strings are composed of markers and data. The markers indicate the type of structure that is being created, and the data represents the parameters of that structure. Some markers are optional, and some are required. Data is encoded in a variety of ways in an attempt to minimize string length. For example, an FBS marker specifies the termination of a path at a raw data location. In order to allow a connection to any raw data location, this value must be encoded exactly. Therefore, the encoding translates directly to a raw data location. As a counterexample, a NAF marker specifies a connection from one neuron to another. This marker is followed by a data value representing the initial weight of this interconnection. Since a connection weight may take on any of the infinite values between -0.5 and $+0.5$, it is impossible to encode all of the possible values. Therefore, a special encoding is used (non-linear gray code) that can encode a discrete subset of the possible values. Note that this encoding is non-linear in order to allow a large range of values.

At this point, it is important to note another capability of this encoding. Since each path created by a string does not have to share nodes with another path, it is possible to have multiple independent networks that combine only at the output node. In this way, it is possible to create network ensembles that identify unique features of the data. This is believed to be one of the major strengths of this system.

3.3.2. Population Creation

The first task of the CEP is to create a population of recognition network strings. The number of recognition network strings that compose the initial population is configurable.

Also, they can be created randomly, or they can be imported from other CEPs. Both of these options are subsequently discussed.

Each random recognition network string that is created can be uniquely configured using a guided or unguided random generation process. The term guided implies that certain features of the encoded networks were controlled so as to improve the likelihood that useful networks would be created. For example, one attribute that is controlled is the interconnection of ANN nodes. A fully random generation process could create ANNs with sparse interconnections between neurons. These types of networks are generally less effective than fully interconnected networks. Thus, by constraining the generation process to create more highly interconnected networks, the initial population will have higher average fitness, and evolution will converge on successful solutions more rapidly.

The initial population can also contain imported networks. These networks are usually the successful elements of previous CEP runs. This ability to further evolve the successes of other CEP runs is important in exploiting the parallelism of evolution. Since the CEP program was not designed to use processor threading, it is a sequential process. However, by providing this importation capability, it allows for multiple CEP programs to be run in parallel, succeeded by a secondary run that combines and further evolves all of the successful recognition networks from previous runs.

3.3.3. Population Evaluation

Once a population of recognition network strings is created, it is then necessary to do an initial evaluation of every network in the population. This evaluation process consists of two steps: network implementation and network evaluation.

Network implementation involves decoding the string and creating the corresponding network in memory. This process is relatively fast, so only one network is kept in memory at a time to maximize the resources available for the evaluation of that network.

Network evaluation involves triggering each of the input neurons with the application data and evaluating the result after the transactions have propagated to the output neurons. The success of each evaluation is recorded and the overall fitness of the network is determined. This fitness data is stored along with the string for future reference.

3.3.4. Evolution

After determining the fitness of each population member, the task of evolution begins. This process consists of five steps: parent selection, offspring replacement selection, parent crossover, offspring mutation, and offspring evaluation. Each of these steps are repeated for a specified number of times or until a successful solution is found.

3.3.4.1. Parent Selection

The first step of the evolutionary process is the selection of eligible parents from the population using a contest algorithm. Two parents are selected at random to begin. Next, a replacement is selected for each parent at random. If the new selection has a higher fitness than the previous selection, then the previous selection is discarded. Otherwise the new selection is discarded. This process is repeated for each parent a specified number of times in order to probabilistically find the highest fitness networks in the population. Note that this process does not require implementation or evaluation of the networks, because it merely uses the fitness number that was associated with each string during the population creation phase.

3.3.4.2. Offspring Replacement Selection

The purpose of this step is to limit the population size by replacing the lowest fitness members of the population with the offspring that are about to be generated. The process of selecting the candidates to be replaced is identical to the parent selection algorithm with the exception that eligible candidates are selected according to lowest fitness, rather than highest fitness.

3.3.4.3. Parent Crossover

This is the exploitation mechanism of the CEP. The two parent strings that were selected in the first step are crossed over to create two offspring such that the parents' genetic material is randomly shared between the children. Each child is individually created by randomly selecting string segments from one of the two parents. Obviously, there are certain rules that regulate this procedure in order to maintain the integrity of the resulting strings. First, string segments can only be copied starting and ending at marker characters. Also, certain markers can only be replaced by a limited selection of other markers.

3.3.4.4. Offspring Mutation

Once offspring have been created from the parents, some mutations are introduced into the offspring. This is the exploration mechanism of the CEP. Certain rules regulate this procedure as well. For example, marker characters cannot be mutated because this would corrupt the string.

3.3.4.5. Offspring Evaluation

The last step in the evolution process involves testing the newly created offspring to determine their fitness. This is the same process as that used to evaluate fitness during population creation.

4. Empirical Analysis

The fundamental assertion of this thesis is that concurrently evolved feature extractors and ANNs are more successful than independently evolved versions. As stated earlier, it is impractical to prove the effectiveness of the concurrent evolutionary approach using theoretical means. Therefore, two experiments, each composed of numerous tests, were conducted in an attempt to provide support for this thesis.

4.1. Testing Methodology

Before describing the specific tests that were conducted, it is necessary to describe the testing environment to provide a frame of reference for understanding the test results. This section addresses this topic.

4.1.1. Application

In order to quantify the efficacy of the evolved systems, it was necessary to select a target application for testing. It was crucial that the selected application be challenging enough to provide meaningful results, yet simple enough to minimize computations that do not specifically relate to the goals of this thesis. Furthermore, the application should be typical of a real-world problem that could be solved using ANNs.

In fulfillment of these requirements, a digit recognition application was selected for all experimentation. A database of 15,000 handwritten digits, representing numbers between zero and nine, was used. Figure 4 shows examples of each digit. The frequency of each digit is shown in Table 2 for the full 15,000 digit database, and Table 3 shows the digit frequency for a 2,000 digit subset of the database that was used for some of the testing. These digits were taken from a variety of writers with a wide range of writing styles.

They are size normalized and contain little or no noise resulting from input processing. However, they have varying stroke widths and are not necessarily aligned within the bounding box. Each digit is represented by a 30x20 binary bitmap followed by a key indicating its correct value (Figure 5). Within the bitmap, zeros indicate the presence of ink, whereas ones indicate whitespace.

Figure 4 – Sample Database Digits

Digit	1	2	3	4	5	6	7	8	9	0
Frequency	1229	1176	1751	1563	716	1799	1859	1680	1814	1413

Table 2 – Frequency of Digits in 15,000 Digit Database

Digit	1	2	3	4	5	6	7	8	9	0
Frequency	209	155	240	207	80	209	257	209	254	180

Table 3 – Frequency of Digits in 2,000 Digit Subset of Database

4.1.1.1. Database Partitioning

15,000-digit Database				
Unused Digits	Active Testing Subset (size varies per test)			Unused Digits
	CEP Training Set (50% of Active Testing Subset)		CEP Testing Set (50% of Active Testing Subset)	
	ANN Training Set (25%)	ANN Testing Set (25%)	ANN Training Set (25%)	ANN Testing Set (25%)

Table 4 – Database Partitioning

Table 4 shows how the database was partitioned. Due to computational limitations, the actual testing set was not the entire 15,000-digit database for most tests. Rather, an active subset was selected, and the remaining digits were left unused. The description of each test indicates the size of the active testing subset that was used. Also note that the training and testing set were divided equally so as not to introduce asymmetry into the testing process.

The CEP training set was used to evolve successful recognition networks. During the evolution process, each network was trained and tested using the ANN training set and the ANN testing set within the CEP training set partition. Once evolution was complete, the successful networks were trained and tested using the ANN training set and ANN testing set within the CEP testing set partition. It was necessary to keep the training and testing sets disjoint in order to get an accurate evaluation of the success of each recognition network. Otherwise, a network could appear successful by merely learning specific features of the application data. This would lead to the evolution of recognition networks that were limited in their generalization ability.

Finally, it is important to quantitatively define how the database was partitioned for the testing. Even though the relative partition sizes are variable in the system, each partition was fixed to be of equal size in order to simplify testing. So, as an example, if the active testing subset is comprised of 1000 digits, then the CEP training and testing sets would each be 500 digits, and the ANN training and testing sets therein would each be 250 digits.

4.1.2. Fitness Evaluation

At this point, it is necessary to address what fitness means when describing a recognition network. Fitness is measured by the performance of the system when tested with the application data.

Each digit bitmap from both the training and testing sets is applied to the network, and the output of the network for that digit is recorded. The output neuron with the highest value indicates the network's best guess at which digit is present on its input. The second highest output neuron value is then subtracted from the highest one, yielding a confidence value. This confidence value is compared against a confidence threshold, which is evolved with the network. The combination of the network's guess and its confidence value produces four possible categories to describe the network's response as shown in Table 5. The best-case result is a qualified success, while the worst-case result is a qualified error. The basic fitness function used for the majority of testing is shown in Equation 1.

$$Fitness = 10S_{train} + 20S_{test} - E_{train} - 2E_{test} - 5R_{train} - R_{test} - C \quad (1)$$

where S = Unqualified Success Rate, E = Qualified Error Rate, R = Qualified Rejection Rate, C = Network Complexity

The largest positive component of the fitness function is the success rate of the network. Note that this includes both qualified and unqualified successes. On the negative side, the major component that reduces fitness is the qualified error rate of the network. This

is necessary to avoid networks that have high success rates at the expense of incorrectly guessing some digits with high confidence. Also, the rejection rate is taken into consideration so as to prevent the evolution of networks that have high success rates while rejecting most of the samples.

Category	Best Guess Result	Threshold Comparison
Qualified Success	Correct	Confidence > Threshold
Unqualified (Rejected) Success	Correct	Confidence < Threshold
Unqualified (Rejected) Error	Incorrect	Confidence < Threshold
Qualified Error	Incorrect	Confidence > Threshold

Table 5 – Network Result Categories

The specific ratio of each element contained in the fitness function was selected through a trial and error approach. Further experimentation may reveal a more optimal solution but automated search of the fitness function space is outside the scope of this thesis.

Note that the above components are considered for both the training and the testing data. It is important to consider the training data as well in order to evolve networks that can train well. However, the training results are not as heavily weighted as the testing results so that networks are discouraged from just memorizing the training data.

Finally, network complexity is also considered as a negative component of the fitness function. The complexity calculation is shown in Equation 2. This component is necessary so that evolved networks do not get arbitrarily complex, thereby bottlenecking the entire algorithm.

$$NetworkComplexity = (N + Y_N + F + Y_F + T) \times I \quad (2)$$

where N = # Neurons, Y_N = # Neuron Synapses, Y_F = # Feuron Synapses, F = # Feurons, T = # Features, I = # Training Iterations

Note that the complexity is a product of the number of training iterations. This is critical in evolving networks that train quickly.

4.1.3. Testing Parameters

Each experiment that is performed has a handful of parameters that are varied to control the operation of the algorithm. These parameters are listed in table form alongside of each test description. Thus, the meaning of these parameters is subsequently described for reference:

Active Testing Subset Size – The number of digits that were used during this test. The specific partitioning of the digit set is described in section 4.1.1.1.

Initial Population Size – This is the number of recognition networks that are created through various random processes to comprise the initial population for the CEP. The initial population creation process is described in section 3.3.2

Tournament Size – This is the number of contests that are conducted in order to choose high fitness recognition networks for reproduction and low fitness recognition networks for subsequent replacement. This selection process is described in sections 3.3.4.1 and 3.3.4.2.

Iterations – This is the number of times two parents are selected and reproduced to create two offspring that are then mutated and inserted into the population. This is a gauge of how long evolution lasts for a specific test before the test is terminated.

4.2. Testing Results and Analysis

Two experiments were conducted, each consisting of numerous individual tests, in order to support the assertions of this thesis. The motivations for selecting the appropriate tests for each experiment and the parameters for each test are subsequently described along with the corresponding results and analysis.

It is important to note that all graphs display only the results of applying recognition networks to the ANN testing set portion of the CEP testing set. The training set results are purposely omitted because they do not describe the performance of the recognition network on an independent test set. The training set data is merely an intermediate metric used to evaluate the expected performance of the network.

4.2.1. Experiment #1 – Constrained CEP Execution

The first experiment, comprised of four tests, was designed to prove that the concurrent evolution of each recognition network component is beneficial to the search for an optimal solution. To this end, the effect of constraining the evolution of various recognition network components was analyzed. The first test (baseline test) was intended as a frame of reference for subsequent tests. In the remaining tests, the creation and/or evolution of specific recognition network components was constrained and compared to the baseline test to determine the benefit of concurrent evolution of that component.

Due to computational limitations, these tests utilized a limited CEP population size as well as a limited evolutionary period (see parameter tables). Therefore, the recognition networks that are evolved in these tests were below optimal. The purpose of this experiment was not intended to demonstrate the maximum capability of the CEP.

4.2.1.1. Baseline Test

The purpose of the baseline test was to provide a frame of reference for subsequent tests that make targeted modifications of the baseline algorithm. The results of this test are shown in Figure 6. However, they are not significant independently. Therefore, they are superimposed on the graphed results of subsequent tests. Table 6 shows the parameters selected for the baseline test.

Figure 6 - Baseline Test Success Rates

Parameter	Value
Active Testing Subset Size	2000 digits
Initial Population Size	2048 recognition networks
Tournament Size	32
Iterations	1024

Table 6 – Baseline CEP Test Parameters

4.2.1.2. Feature Extractor Contribution Analysis 1

The first assertion that must be proven is that feature extractors offer some benefit to evolved ANNs. In order to validate this assertion, it was necessary to analyze the effect of removing the feature extractor component from the evolutionary process. To this end, a test was conducted wherein the baseline CEP was modified to eliminate feature extractors altogether from recognition networks and to apply the digit bitmap data directly to the ANN inputs. These evolved networks were still able to select which bitmap features to utilize, but they could not insert functional components (feurons) between the bitmap features and the ANN inputs.

The results of this test have been broken down into two graphs. Figure 7 illustrates the unqualified success rate of the constrained CEP compared to that of the baseline test. It is clear that the constrained CEP struggled to reach even 50% recognition. Since a purely random guesser would likely achieve recognition rates around 10%, this confirms that the ANN is functioning properly, but is hindered due to the missing functional nodes.

Figure 8 shows that the CEP has difficulty finding an optimal rejection rate that allows the qualified success rate to continually improve. This is likely due to the noise present in the input data due to the direct mapping to pixel locations.

Clearly, the insertion of feurons that perform mathematical operations on the digit bitmap prior to ANN application is effective at improving the success rate of evolved recognition networks.

Figure 7 - Missing Feature Extractor Test vs. Baseline Test
Unqualified Success Rates

Figure 8 – Missing Feature Extractor Test vs. Baseline Test
Qualified Success Rates

4.2.1.3. Feature Extractor Contribution Analysis 2

Although the degradation of performance due to the removal of feature extractors altogether demonstrates that they do offer some benefit to recognition network performance, it does not prove that the concurrent evolution of feature extractors and ANNs is beneficial. In order to validate this assertion, it was necessary to allow the inclusion of feature extractors in the recognition networks comprising the initial population while restricting the ability of this portion of each recognition network to evolve. To accomplish this, the baseline CEP algorithm was modified so as not to allow feature extractors to be involved in the crossover operation. This still permitted random search through the feature extractor space, but prevented hill-climbing improvements. Note that the random search was achieved not only by creating a random initial population, but also through use of the mutation operator during evolution.

Figure 9 and Figure 10 illustrate the results of this test relative to the first feature extraction contribution test. It is clear that the addition of feature extractor components is beneficial to the performance of the system. This is an expected result because some amount of feature extraction is necessary to simplify the job of the ANN. However, as indicated by Figure 11 and Figure 12, the performance does not match that of the baseline test due to the inability of the feature extractors to concurrently evolve with the ANNs. These findings provide support to the argument that the evolution of the feature extractors adds value to the evolved recognition systems.

Figure 9 - Constrained Feature Extractor Test vs. Missing Feature Extractor Test
Unqualified Success Rates

Figure 10 - Constrained Feature Extractor Test vs. Missing Feature Extractor Test
Qualified Success Rates

Figure 11 - Constrained Feature Extractor Test vs. Baseline Test
Unqualified Success Rates

Figure 12 - Constrained Feature Extractor Test vs. Baseline Test
Qualified Success Rates

4.2.1.4. ANN Contribution Analysis

The previous feature extractor contribution experiments demonstrated that some benefit was offered by evolving feature extractors. However, they did not prove that evolving the ANN component was beneficial to the process. Therefore, another test was performed where the baseline test was modified to inhibit evolution of the ANN portion of the recognition networks. This was accomplished by not allowing ANNs to be involved in the crossover operation. This still permitted random search through the ANN customization space, but prevented hill-climbing improvements. Note that the random search was achieved not only by creating a random initial population, but also through use of the mutation operator during evolution.

Figure 13 and Figure 14 illustrate the results of this test. Although some performance degradation is evident, it does not appear to be as severe as in previous tests. This is likely because the ANN search space is not nearly as large as the feature extractor search space. Since the CEP had a large selection of ANNs available in the initial population, it was probably able to find one that approximated the optimal ANN. This could be due to the fact that an overly complex ANN is likely to achieve success rates similar to a smaller, more optimized ANN given sufficient training iterations.

Figure 13 - Constrained ANN Test vs. Baseline Test
Unqualified Success Rates

Figure 14 - Constrained ANN Test vs. Baseline Test
Qualified Success Rates

4.2.2. Experiment #2 – Evolution of Optimal Recognition Networks

Due to computational limitations, the CEP was constrained in the first experiment and was therefore unlikely to evolve optimal solutions. Consequently, a second experiment was performed that was intended solely to demonstrate the capabilities of the CEP.

4.2.2.1. Evolutionary Process Analysis

This experiment was conducted with a large initial population and a long, three stage evolutionary period as shown in Figure 15. By partitioning the evolutionary period into three stages, it was possible to take advantage of the parallelism inherent in EPs. The first stage consisted of 16 independent CEP runs, each evolving a random initial population. The CEP runs for the second and third stages did not use a random initial population. Rather, their initial population was comprised of the highest fitness members created by the CEP runs from the previous stage. As such, the second stage reduced the number of CEP runs from sixteen down to four such that each CEP combined the top recognition networks from four of the first stage CEP runs. Similarly, the third stage used the top recognition networks created by the second stage CEP runs to create a final population of highly fit recognition networks. The specific parameters used for each stage are detailed in Table 7, Table 8, and Table 9.

The highest fitness member success rates for each stage of evolution are shown in Figure 16, Figure 17, and Table 10. Also, Figure 18 is a scatter plot of the recognition network population fitness at each stage of evolution (only one population from stage 1 is shown for clarity). Note that the step pattern visible at the bottom of the stage 1 data is due to the process by which the initial random population was generated. For each of the first stage CEP runs, half of the initial members were created with a random generation process while the other half were created with a guided generation process, thereby slightly increasing the fitness of these members.

Figure 15 – Optimal Recognition Network Experiment Flow

Parameter	Value
Active Testing Subset Size	2000 digits
Initial Population Size	2048 recognition networks
Tournament Size	8
Iterations	512

Table 7 – Stage 1 CEP Parameters

Parameter	Value
Active Testing Subset Size	2000 digits
Initial Population Size	1024 recognition networks
Tournament Size	16
Iterations	1024

Table 8 – Stage 2 CEP Test Parameters

Parameter	Value
Active Testing Subset Size	2000 digits
Initial Population Size	1024 recognition networks
Tournament Size	16
Iterations	1024

Table 9 – Stage 3 CEP Test Parameters

Figure 16 – Evolution of Optimal Networks Test Unqualified Success Rates

Figure 17 – Evolution of Optimal Networks Test Qualified Success Rates

Stage	Unqualified Success Rate	Qualified Success Rate	Qualified Rejection Rate
1	82%	84.19%	7.6%
2	92.4%	93%	5%
3	93.6%	95.16%	4.8%

Table 10 – Maximum Recognition Rates for Optimal Network Test

Figure 18 – Population Fitness at Each Stage of Evolution

The first stage of evolution produced a wide range of success rates. The highest fitness member from all 16 of the first stage CEP runs achieved a success rate of 84.19% with a rejection rate of 7.6% or 82% with 0% rejection. However, a few of the 16 first stage runs generated only poor recognition networks. One run was only able to evolve a recognition network that achieved a 46.54% recognition rate with 21.8% rejection or 41% with 0% rejection. This result is only a slight improvement over the best recognition

rates found within the initial random population. Note that the number of evolutionary iterations in this stage was small compared to the initial population size. This was intentional so as to leave a relatively diverse population for the second stage. The large variation in fitness values of recognition networks created by the first stage (Figure 18) illustrates this diversity.

The second stage of evolution generated substantially higher fitness members. The highest recognition network demonstrated recognition rates of 93% with a 5% rejection rate or 92.4% with a 0% rejection rate. This is already inline with results achieved in other research described earlier. However, the rate at which network fitness improved substantially decreased relative to the first stage evolution. This is an expected result as recognition rates approach the 100% bound.

After three stages of evolution, starting with a total initial population of 32,768 random recognition networks, the maximum fitness member demonstrated recognition success rates of 95.16% with a 4.8% rejection rate or 93.6% with a 0% rejection rate. Although this is the best result obtained in this specific test, it is certain that better results can be obtained by applying additional computing resources to this task. In particular, the initial population size and evolution length parameters are unbounded and further increases of these parameters would probably improve results. Furthermore, a variety of other CEP parameters could also be adjusted that may improve the final result.

Another important factor to consider when evaluating the effectiveness of the best recognition network is the limitations of the digit database. It is readily evident that the use of larger training and testing sets could improve the performance of the evolved networks. In fact, by retraining and testing the highest fitness recognition network using the full 15,000 digit database yielded a success rate of 95.7% with 4.5% rejection or 93.6% with a 0% rejection rate. Of course, since this network was evolved using a much smaller training set, it is probably not optimally configured for a longer learning process across more data samples. Therefore, it is likely that this recognition rate could be

further improved if the larger database was used throughout the evolutionary process. However, computational and schedule limitations prevented such an experiment.

4.2.2.2. Successful Recognition Network Properties Analysis

Now that a highly successful recognition network has been created, it makes sense to analyze the properties of this network. Of course, a complete deconstruction of the function network algorithm and ANN parameters is unrealistic due to the complexity of the recognition network. Nevertheless, some attempt should be made to understand its topology and behavior.

4.2.2.2.1. Recognition Network Computational Requirements

Table 11 demonstrates the performance of the maximum fitness recognition network on a 2.2 GHz Intel Pentium IV processor running Linux. These times translate to a processing rate of approximately 68 digits / second. Although this speed is impractical for commercial applications, it should be noted that a commercial implementation would require deconstruction of the evolved algorithm followed by hardware implementation to improve performance. In fact, the cost of hardware implementation for the evolved functions could be a factor in the fitness equation in order to evolve hardware friendly algorithms. Furthermore, the hardware implementation would reflect the recognition network after training, thereby eliminating any need to incur the training expense.

Task	Time
Network Creation	< 1 second
Network Training on 15,000 Digits	128 minutes 26 seconds
Network Evaluation of 15,000 Digits	3 minutes 40 seconds

Table 11 – Time Requirements for Recognition Network Tasks

4.2.2.2.2. ANN Properties

The ANN portion of the highest fitness recognition network consists of 33 neurons with 232 inter-node connections and 1,985 connections to feuron nodes. The ANN has a maximum of five layers of depth and a maximum width of 14 nodes. The network was evolved to use 468 training iterations to learn the training data. The confidence threshold used to accept or reject classifications was evolved to 0.5.

4.2.2.2.3. Function Network Properties

The function network portion of the highest fitness recognition network consists of 84 feurons with 143 inter-node connections of which 76 connect directly to an input feuron. The functions that were used within this network include multiplication, addition, averaging, comparisons, absolute value and cosine.

4.2.2.2.4. Classification Error Analysis

Finally, it is necessary to analyze the data that is being misclassified in order to understand the performance of the recognition network. Figure 19 shows some of the digits that were misclassified and rejected by the recognition network due to excessive error. Figure 20 shows some of the digits that were misclassified but accepted by the recognition network because the output error was within the threshold level. Note that the digits that appear to be cropped in these figures are actually cropped in the original database, which is likely the cause of their misidentification.

It is difficult to find the reason for misclassification in many cases, while others are obvious. Interestingly, there was not one particular digit that was consistently misclassified, but erroneous classification occurred uniformly with all digits. Also, when digits were misclassified, the recognition network guesses did not skew towards one particular digit.

Although it is desirable to improve recognition network performance by finding patterns of misclassification, this proves to be a fruitless task due to the seemingly random mistakes of the recognition network. Furthermore, it should be noted that misclassifications might result not only from defects in the recognition network logic, but also because of problems with the dataset. Dataset inconsistencies may be even harder to identify.

Figure 19 – Rejected Digit Errors (A = Actual Digit, G = Best Guess)

Figure 20 – Accepted Digit Errors (A = Actual Digit, G = Best Guess)

5. Conclusions and Further Research

This thesis describes a novel approach for generating recognition systems through concurrent evolution of feature extractors and artificial neural networks. This method was tested under a variety of constraints to validate its effectiveness. The results obtained from testing indicate that the recognition systems created using this method have similar performance characteristics to other approaches found in research literature. However, the main advantage of this approach is that it does not require expert knowledge about the problem domain to generate effective solutions. Therefore, it can easily be applied to other problems where expert knowledge is costly or not readily available.

Although the concurrent evolutionary process designed for this research performed reasonably well, numerous improvements can still be made. First and foremost, some of the algorithms used to implement the recognition networks are somewhat inefficient, so a variety of speedups are possible. Second, very little effort was made to limit the creation of inefficient feature extraction networks, resulting in the dilution of the CEP's population pool with low fitness members. This kind of problem can be addressed by redesigning the network representation scheme (genome mapping), which could offer additional benefits as well.

Aside from specific improvements to the CEP algorithms, the next logical step is to apply this method to a number of other problem domains to evaluate its general applicability. A variety of newly emerging recognition tasks such as face recognition or scene analysis would be appropriate because of the limited expertise available on these topics. The major challenge in applying a CEP to these tasks would be the computational requirements to process the large amount of data in these complex domains. However, with efficient design and faster processors, the exploration of CEPs for these domains is completely realistic.

Finally, it should be noted that this thesis focused on providing solutions to problem domains where expert knowledge was unavailable or costly. However, if some expert knowledge is available, the expert algorithms can be incorporated into the initial population of the CEP in order to improve the performance of the networks. The analysis of this technique also merits further research.

References

- [LeCun89] Le Cun, Y., Jackel, L. D., Boser, B., and Denker, J. S., Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning, IEEE Communications Magazine, vol. 27, no. 11, pp. 41-46, 1989.
- [Hannak01] V. Hannak, Pattern Recognition Term Project: Voice Recognition System, Rochester Institute of Technology, Rochester, NY, 2001.
- [Lathi92] B. P. Lathi, Linear Systems and Signals, Berkeley Cambridge Press, 1992.
- [Goldberg89] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1989.
- [Bownds99] M. D. Bownds, The Biology of the Mind: Origins and Structures of Mind, Brain, and Consciousness, Fitzgerald Science Press, 1999.
- [Ash89] T. Ash, Dynamic Node Creation, Connection Science, vol. 1, no. 4, pp. 365-375, 1989.
- [Fahlman91] S. E. Fahlman and C. Lebiere, The Cascade-Correlation Learning Architecture, Advances in Neural Information Processing Systems, vol. 2, (D. S. Touretzky, ed.), pp. 524-532, Morgan Kaufmann, San Mateo, CA, 1990.
- [Setiono95] R. Setiono and L. C. K. Hui, "Use of a Quasi-Newton Method in a Feedforward Neural Network Construction Algorithm, IEEE Transactions on Neural Networks, vol. 5, no. 1, pp. 273-277, 1995.
- [Nadal89] J. Nadal, Study of a Growth Algorithm for a Feedforward Network, International Journal of Neural Systems, vol. 1, pp. 55-59, 1989.

- [Burgess94] N. Burgess, A Constructive Algorithm that Converges for Real-Valued Input Patterns, *International Journal of Neural Systems*, vol. 5, no. 1, pp. 59-66, 1994.
- [Reed93] R. Reed, Pruning Algorithms - A Survey, *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740-747, 1993.
- [Angeline94] P. J. Angeline, G. M. Sauders, and J. B. Pollack, An Evolutionary Algorithm that Constructs Recurrent Neural Networks, *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 54-65, 1994.
- [Miller89] G. F. Miller, P. M. Todd, and S. U. Hegde, Designing Neural Networks Using Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms and their Applications* (J. D. Schaffer, ed.), pp. 379-384, Morgan Kaufmann, San Mateo, CA, 1989.
- [Yao97] X. Yao and Y. Liu, A new evolutionary system for evolving artificial neural networks, *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694--713, 1997.
- [Maniezzo94] V. Maniezzo, Genetic Evolution of the Topology and Weight Distribution of Neural Networks, *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 39-53, 1994.
- [Fogel95] D. B. Fogel, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, IEEE Press, New York, NY 10017-2394, 1995.
- [Liu96] Y. Liu and X. Yao, Evolutionary Design of Artificial Neural Networks with Different Nodes, *Proceedings of the 1996 IEEE Int'l Conference on Evolutionary Computation (ICEC'96)*, Nagoya, Japan, pp. 670-675, IEEE Press, New York, NY 10017-2394, 1996.

- [Liu97] Y. Liu and X. Yao, Evolving Modular Neural Networks which Generalise Well, Proceedings of the IEEE Conference on Evolutionary Computation, pp. 605-610, 1997
- [Yao93] X. Yao, Evolutionary Artificial Networks, International Journal of Neural Systems, vol. 4, no. 3, pp. 203-222, 1993.
- [Balakrishnan95] K. Balakrishnan and V. Honavar, Properties of Genetic Representations of Neural Architectures, Proceedings of the World Congress on Neural Networks, pp. 807-813, 1995.
- [Darwen96] P. Darwen and X. Yao, Automatic Modularization by Speciation, Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), Nagoya, Japan, pp. 88-93, IEEE Press, New York, NY 10017-2394, 1996.
- [Gruau95] F. Gruau, Automatic Definition of Modular Neural Networks, Adaptive Behavior, vol. 3, no. 2, pp. 151-183, 1995.
- [Hancock92] P. J. B. Hancock, Genetic Algorithms and Permutation Problems: A Comparison of Recombination Operators for Neural Net Structure Specification, Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92) (D. Whitley and J. D. Schaffer, eds.), pp. 108-122, IEEE Computer Society Press, Los Alamitos, CA, 1992.
- [Schaffer92] J. D. Schaffer, D. Whitley, and L. J. Eshelman, Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art, Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92) (D. Whitley and J. D. Schaffer, eds.), pp. 1-37, IEEE Computer Society Press, Los Alamitos, CA, 1992.
- [Belew91] R. K. Belew, J. McInerney, and N. N. Schraudolph, Evolving Networks: Using Genetic Algorithm with Connectionist Learning, Computer Science and

Engineering Department Technical Report #CS90-174, University of California at San Diego, La Jolla, CA 92093, 1991.

[Sharkey96] A. J. C. Sharkey, On Combining Artificial Neural Nets, Connection Science, vol. 8, pp. 299-313, 1996.

[Hashem93] S. Hashem, Optimal Linear Combinations of Neural Networks, Ph.D. Thesis, Purdue University, 1993.

[Sharkey97] A. J. C. Sharkey and N.E. Sharkey, Combining Diverse Neural Nets, The Knowledge Engineering Review, vol. 12, no. 3, pp. 231-247, 1997.

[Opitz96] D. W. Opitz and J. W. Shavlik, Generating Accurate and Diverse Members of a Neural-Network Ensemble, Advances in Neural Information Processing Systems, vol. 8, pp. 535-543, 1996.

[Lawrence97] S. Lawrence, C. L. Giles, A. C. Tsoi and A. D. Back, Face Recognition: A Convolutional Neural Network Approach, IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition, vol. 8, no. 1, pp. 98-113, 1997.

[Neubauer98] C. Neubauer, Evaluation of Convolutional Neural Networks for Visual Recognition, IEEE Transactions on Neural Networks, vol. 9, no. 4, 1998.

[Towell90] G. G. Towell, J. W. Shavlik and M.O. Noordewier, Refinement of approximate domain theories by knowledge based neural networks, Proceedings of the Eighth National Conference on Artificial Intelligence, vol. 2, pp. 861-866, 1990.

[Opitz97] D. W. Opitz and J. W. Shavlik, Connectionist Theory Refinement: Genetically Searching the Space of Network Topologies, Journal of Artificial Intelligence Research, vol. 6, pp. 177-209, 1997.

[Burns98] B.D. Burns, INDiGENT: Genetically Refining Expert Neural Networks, B. A. Thesis, 1998.

[Liu98a] H. Liu, Feature Transformation and Subset Selection, Discovery Science, pp. 279-290, 1998.

[Zupan98] B. Zupan, M. Bohanec, J. Demsar, and I. Bratko, Feature Transformation by Function Decomposition, IEEE Intelligent Systems, pp. 38-43, March/April 1998.

[Scheunders98] P. Scheunders, S. De Backer, and A. Naud, Non-linear Mapping for Feature Extraction, SSPR/SPR, pp. 823-830, 1998.

[DeBacker98] S. De Backer, A. Naud, and P. Scheunders, Non-linear Dimensionality Reduction Techniques for Unsupervised Feature Extraction, Pattern Recognition Letter, vol. 19, pp. 711-720, 1998.

[Devijver82] P. Devijver and J. Kittler, Pattern Recognition: A Statistical Approach, Prentice/Hall, Englewood Cliffs, New Jersey, 1982.

[Langley94] P. Langley, Selection of Relevant Features in Machine Learning, Proceedings of the AAAI Fall Symposium on Relevance, New Orleans, LA, pp. 140-144, AAAI Press, 1994.

[Liu98b] H. Liu and R. Setiono, Incremental Feature Selection, Applied Intelligence, vol. 9, no. 3, pp. 217-230, 1998.

[Chang90] E. Chang, R. Lipmann, and D. Tong, Using Genetic Algorithms to Select and Create Features for Pattern Classification, Proceedings of the IEEE International Joint Conference on Neural Networks, vol. 3, pp. 747-752, 1990.

[Firpi99] H. Firpi and J. Echauz, Genetically Found, Neurally Computed Artificial Features, Computing Research Conference, Mayaguez, Puerto Rico, 1999.

[Gaborski93] R. Gaborski, P. Anderson and D. Tilley, Genetic Algorithm Selection for Handwritten Character Identification, International Conference on Neural Networks and Genetic Algorithms, Innsbruck, Austria, 1993.

[Russell95] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995.

[Mori92] S. Mori, C. Y. Suen, and K. Yamamoto, Computer Recognition of Unconstrained Handwritten Numerals, Optical Character Recognition, Special Issue of Proceedings of the IEEE, (T. Pavlidis and S. Mori, eds.), vol. 80, no. 7, pp. 1162-1180, July 1992.

[Cherkassky98] V. Cherkassky and F. Mulier, Learning from Data: Concepts, Theory, and Methods, Wiley, NY, 1998.

[Breuel93] T. M. Breuel, Recognition of Handprinted Digits Using Optimal Bounded Error Matching, IDIAP, Memo #93-06, 1993.

[Alt62] F. L. Alt, Digital Pattern Recognition by Moments, Optical Character Recognition (G. L. Fischer, ed.), pp. 159-179, McGreger & Werner, 1962.

[Casey70] R. G. Casey, Moment Normalization of Handprinted Characters, IBM Journal of Research and Development, vol. 14, pp. 548-557, 1970.

[Knerr92] S. Knerr, L. Personnaz, and G. Dreyfus, Handwritten Digit Recognition by Neural Networks with Single Layer Training, IEEE Transactions on Neural Networks, vol. 3, no. 6, pp. 962-968, 1992.

- [Behnke97] S. Behnke, M. Pfister and R. Rojas, Recognition of Handwritten Digits Using Structural Information, Proceedings ICNN'97, Houston, pp. 1391-1396, 1997.
- [Lai81] M. T. Y. Lai and C. Y. Suen, Automatic Recognition of Characters by Fourier Descriptors and Boundary Encoding, Pattern Recognition, vol. 14, no. 1-6, pp. 383-393, 1981.
- [Belkasim91] S. O. Belkasim, M. Shridhar, M. Ahmadi, Pattern Recognition with Moment Invariants: A Comparative Study and New Results, Pattern Recognition, vol. 24, no. 12, pp. 1117-1138, 1991.
- [Cash87] G. L. Cash and M. Hatamian, Optical Character Recognition by the Method of Moments, Computer Vision, Graphics, and Image Processing (CVGIP), vol. 39, pp. 291-310, 1987.
- [Duda00] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification, Second Edition, Wiley, 2000.
- [Grother93] P. J. Grother and G. T. Candela, Comparison of Handprinted Digit Classifiers, Technical Report NISTIR 5209, NIST, 1993.
- [Denker89] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I. Guyon, Neural Network Recognizer for Hand-Written Zip Code Digits, Advances in Neural Information Processing Systems (D. Touretzky, ed.), vol. 1, pp. 323-331, Morgan Kaufmann, 1989.
- [LeCun90] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Handwritten Digit Recognition with a Back-Propagation Network, Advances in Neural Information Processing Systems (D. Touretzky, ed.), vol. 2, pp. 598-605, Morgan Kaufmann, 1990.

[Behnke98] S. Behnke, M. Pfister and R. Rojas, A Study on the Combination of Classifiers for Handwritten Digit Recognition, Proceedings NN'98, pp. 39-46, 1998.

[Vuurpijl98] L. Vuurpijl and L. Schomaker, Multiple-Agent Architectures for the Classification of Handwritten Text, Submitted to IWFHR6, International Workshop on Frontiers of Handwriting Recognition, 1998.

[Hansen97] L. K. Hansen, C. Liisberg, and P. Salamon, The Error-Reject Tradeoff, Open Systems & Information Dynamics, vol. 4, pp. 159-184, 1997.

[Suen92] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam, Computer Recognition of Unconstrained Handwritten Numerals, Optical Character Recognition, Special Issue of Proceedings of the IEEE, (T. Pavlidis and S. Mori, eds.), vol. 80, no. 7, pp. 1029-1058, July 1992.

[Duerr80] B. Duerr, W. Haettich, H. Tropf, and G. Winkler, A Combination of Statistical and Syntactical Pattern Recognition Applied to Classification of Unconstrained Handwritten Numerals, Pattern Recognition, vol. 12, pp. 189-199, 1980.

Glossary

ANN – see artificial neural network

artificial neural network – a programmatic network of parallel processing nodes modeling the function of the human brain

CEP – see concurrent evolutionary process

concurrent evolutionary process – an evolutionary process by which data pre-processing nodes are concurrently evolved with artificial neural networks

EP – see evolutionary process

evolutionary process – a programmatic process modeling natural evolution to search for optimal solutions

feuron – a single processing element in a data pre-processing / feature extraction network

neuron – a single processing element in an artificial neural network (see artificial neural network)