

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of master's thesis by

OLEKSANDR KOSOLAPOV

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Dr. Christopher G. Prince

Name of Faculty Adviser

Signature of Faculty Adviser

Date

GRADUATE SCHOOL

**The Effects of Category Information on Association Learning Tasks
in Neural Network Models**

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA

BY

Oleksandr Kosolapov

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

October 2003

Acknowledgements

I would like to thank my adviser, Dr. Christopher Prince, for his patience, opportunity to participate in his research projects and his guidance of my work. I would like to thank my committee, Dr. Ted Pedersen and Dr. Robert McFarland, for their feedback and suggestions. Also I would like to thank Dr. Yongcheng Qi for his feedback and suggestions. Finally, I would like to thank University of Minnesota Duluth Computer Science Department Faculty and Staff for giving me an honor of studying here.

Abstract

In this thesis we explore the effects of category information on association learning in computational models of neural networks. In order to investigate a range of learning outcomes, we used both simultaneous and sequential learning tasks. A simultaneous learning task involves training on all data at the same time. A sequential learning task involves first training on initial data, and then training on another set of data. In neural networks a typical outcome of sequential learning is a substantial performance reduction on the initial training set, referred to as catastrophic interference (e.g., Ratcliff, 1990). We hypothesized that providing explicit category information as an input when learning associations may improve performance in our learning models. To assess association learning performance we used criteria of training time, generalization and interference (in the sequential task). Experiments showed that explicit category information may actually cause an increase in interference in sequential learning tasks. However, in both learning tasks category information was found, in some cases, to reduce generalization error and training time.

Contents

Chapter 1. Introduction.....	1
1.1 Introduction.....	1
1.2 Literature Review	2
1.2.1 Association Learning Models	3
1.2.2 Category Learning Models.....	6
1.2.3 Interference Reduction Techniques	10
1.3 Overview of Experiments	17
Chapter 2. Experiment 1	20
2.1 Introduction.....	20
2.2 Methods	25
2.2.1 Neural Network Architecture.....	25
2.2.2 Data Sets	27
2.2.3 Performance Measures.....	28
2.3 Results.....	28
2.4 Discussion.....	32
Chapter 3. Experiment 2A: A More Complex Learning Task.....	44
3.1 Introduction.....	44
3.2 Methods	47
3.2.1 Neural Network Architecture.....	47
3.2.2 Data Sets	49

3.2.3 Performance Measures.....	50
3.3 Results.....	51
3.4 Discussion.....	56
Chapter 4. Experiment 2B: A Sequential Learning Task.....	69
4.1 Introduction.....	69
4.2 Methods	72
4.2.1 Neural Network Architecture.....	72
4.2.2 Data Sets	73
4.2.3 Performance Measures.....	74
4.3 Results.....	75
4.4 Discussion.....	80
Chapter 5. Discussion	99
5.1 Introduction.....	99
5.2 Results.....	100
5.3 Future Work.....	103
References.....	106

List of Figures

Figure 1.1 Schema of Association Learning	5
Figure 1.2 Revebrating Architecture with Self Refreshing Memory (From Ans & Rousset, 2000)	19
Figure 2.1 Model ₁ Schema of Association Learning	20
Figure 2.2 Example of Everyday Associations	21
Figure 2.3 Computational Example of Association Learning	22
Figure 2.4 Model ₂ Schema of Association Learning.....	24
Figure 2.5 Computational Example of Association Learning With Category Information.....	35
Figure 2.6 Network Architecture	36
Figure 2.7 Amount of Training Required for All Architectures.....	37
Figure 2.8 Amount of Training Required for Zero Category Units Architecture.....	38
Figure 2.9 Generalization Performance for All Architectures.....	39
Figure 2.10 Generalization Performance by Category for Zero Category Units Architecture.....	40
Figure 2.11 Generalization Performance by Category for One Category Unit Architecture.....	41
Figure 2.12 Generalization Performance by Category For Zero Category Units Architecture Simulation with 200,000 Training Epochs Limit and 0.1 Learning Rate.....	43

Figure 3.1 Schema of Associations to be Learned in Experiment 2	59
Figure 3.2 Amount of Training Required for All Architectures	60
Figure 3.3 Amount of Training Required for All Architectures (for all data points)	61
Figure 3.4 Per Sample Error for All Architectures (Training Results).....	62
Figure 3.5 Per Sample Error Terms for Zero Category Units Architecture (Training Results)	63
Figure 3.6 Per Sample Error, ε , for All Architectures (Generalization Tests).....	64
Figure 3.7 $\varepsilon_{density}$ for All Architectures (Generalization Tests)	65
Figure 3.8 $\varepsilon_{misplacement}$ for All Architectures (Generalization Tests)	66
Figure 3.9 ε_{data} for All Architectures (Generalization Tests)	67
Figure 3.10 ε by Category for Zero Category Units Architecture (Generalization Tests)	68
Figure 4.1 Amount of Initial Training (Step B) Required for All Architectures (Case α)	84
Figure 4.2 Amount of Initial Training (Step B) Required for All Architectures (Case β)	85
Figure 4.3 Amount of Subsequent Training (Step D) Required for All Architectures (Case α)	86
Figure 4.4 Initial (Step B) and Subsequent (Step D) Training Time for Zero Category Units Architecture (Case α)	87
Figure 4.5 Initial (Step B) and Subsequent (Step D) Training ε for Zero Category Units Architecture.....	88

Figure 4.6 Initial Training (Step B) and Recall (Step E) ε for Zero Category Units Architecture (Case α)	89
Figure 4.7 Recall (Step E) ε for Zero Category Units Architecture (Case α).....	90
Figure 4.8 Recall (Step E) ε for One Category Unit Architecture (Case α)	91
Figure 4.9 Recall (Step E) ε for All Architectures (Case α)	92
Figure 4.10 Recall (Step E) ε for All Architectures (Case β)	93
Figure 4.11 Initial Generalization Test (Step C) ε for All Architectures (Case α)	94
Figure 4.12 Recall Generalization Test (Step F) ε for All Architectures (Case α)	95
Figure 4.13 Initial (Step C) and Recall (Step F) Generalization Test ε_{data} for All Architectures (Case α)	96
Figure 4.14 Initial (Step C) and Recall (Step F) Generalization Test $\varepsilon_{density}$ for All Architectures (Case α)	97
Figure 4.15 Recall Generalization Test (Step F) $\varepsilon_{misplacement}$ for All Architectures (Case α)	98

List of Tables

Table 2.1 Probabilities of Successful Outcomes of Zero Category Units Architecture by Chance.....	42
--	----

Chapter 1. Introduction

1.1 Introduction

Humans learn associations in various domains. For example, word-learning in language includes relating spoken words to referents and, to some degree, these word-referent relations can be viewed as associations (e.g., Hollich, Hirsh-Pasek & Golinkoff, 2000). When learning associations, it is often helpful for people to make use of category information. For example, we need to distinguish (i.e., categorize) the sound of words, or human speech, from other sounds (e.g., doors slamming) and we also need to utilize this distinction when associating these sounds with other information. Additionally, infants distinguish food on the basis of color and texture, whereas they distinguish tools by shape and rigidity (Shutts & Markson, 2003). Presumably, infants may also utilize this category information when learning associations.

One way to better understand association learning and categorization is through the use of computational modeling. A range of techniques have been forwarded that can enable such modeling. For example, multi-layer feed-forward neural networks have been used to model association learning (Rumelhart, Hinton, & Williams, 1986), and Adaptive Resonance Theory (ART) neural networks (Carpenter & Grossberg, 1987) have been used to model categorization. Multi-layer feed-forward neural networks learn to associate input patterns with output patterns. In this case, input patterns are vectors of real numbers, and output vectors are also vectors of real numbers. ART networks enable

modeling of category learning where the categories are not known *a priori*. They can group similar input patterns to produce specific output patterns for those groups.

In this thesis we explore how category information affects association learning in computational models of neural networks. As category information appears to have benefits in human learning, we hypothesize that category information may have benefits in association learning in neural networks models. We analyze how providing category information to computational association learning methods affects their performance using criteria of training time, generalization and interference. Interference is said to occur when the learning of one task changes the performance of the model or system on a previously learned task. We hypothesize that providing additional category information to a neural network may help it to better distinguish between categories of input patterns and leave a larger scope of possible adjustments to capture other properties of input patterns. This may also lead to reduction in interference between associations from different categories. Reducing interference may be particularly important because neural networks are known to have strongly destructive interference effects known as catastrophic interference (Ans & Rousset, 2000; McClelland, McNaughton & O'Reilly, 1995; Ratcliff, 1990).

1.2 Literature Review

In the rest of this chapter we review literature on topics concerning categorization and association learning techniques.

1.2.1 Association Learning Models

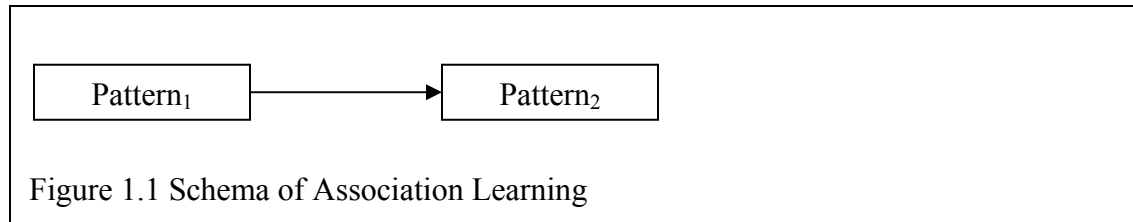
Neural networks may be used for learning associations (Fausett, 1994; Mitchell, 1997). Neural networks have layers of units that act analogously to biological neurons. For example, a unit has an activation value which is a function of units connected to it, and the unit may be connected to other units with different strengths of connections. Feedforward neural networks that have an input layer of units fully connected to an output layer of units are called *single layer perceptrons*. Each connection between units has an associated real valued weight. As there are connections only from input layer to output layer, this corresponds to a single layer of weights. When presented with an input pattern, input units take on activation values that correspond to the values of the input pattern. Output units take as inputs the sum of products of weights and activations of all units connected to the output unit. Various functions (i.e., threshold, sigmoid, etc.) may be used as activation functions for output units. Activations of output units are the network's output. Differences between the network's output and the target or expected output for a particular input pattern is the *error* of the network for the particular input pattern. When a neural network has multiple output units, the similarity of the target output and the actual output vectors (e.g., Euclidean distance) may be used to compute the error. Training of a single layer perceptron can be accomplished by iteratively updating connection weights until an error criterion or maximum number of weight updates is reached. A weight update can be computed according to Equation 1.1, referred to as the *delta rule* (e.g., Fausett, 1994):

$$\Delta w_{ij} = \alpha(t_j - y_j)x_i \quad (1.1)$$

where Δw_{ij} is the change of weight for a connection between input unit i and output unit j , α is the learning rate, t_j is the expected output of unit j , y_j is the actual output of unit j , and x_i is the input to unit i . Training a neural network is started by initializing the connection weights to random values between -1.0 and 1.0, and then iteratively updating weights using the delta rule. Weight updates may be done after presenting a single training pattern (incremental training) or after presenting all training patterns (batch training). In the latter case, weights are updated by the Δw_{ij} values accumulated for all patterns. A single presentation of all training patterns is called an *epoch*. In testing, a network's output and error are computed in the same way as in training, but no weight updates are made.

When single layer perceptrons are trained with the delta weight update rule, convergence to a solution is guaranteed when the training data is linearly separable and the learning rate is sufficiently small (e.g., p. 89, Mitchell, 1997). However, neural networks may produce better results, even when the data is not linearly separable and contains noise when they have *multiple layers* of weights and use other training algorithms. For example, the backpropagation learning algorithm for multilayer feedforward neural networks (Rumelhart et al., 1986). Multilayer feedforward neural networks have input layers of units which may be fully connected to subsequent layers of *hidden* units. The layers of hidden units are connected to the layer of output units or another layer of hidden units. Using the backpropagation algorithm, activation is propagated forward to the output units, and in training the resulting error between the actual output and target output is propagated backwards through the network layers. Connection weights are updated based on the portion of the error generated by a unit.

Units typically have a sigmoid activation function. Neural networks trained with backpropagation use the gradient descent method in the search for the solution. Although gradient descent does not guarantee a global minimum, backpropagation neural networks with sigmoid activation function can approximate any function with arbitrary high precision (e.g., p. 105, Mitchell, 1997).



Feedforward neural networks may be used to model association learning. By association learning, we mean establishing some relationship between pairs of data patterns. We may represent real world data in a form suitable for processing by neural networks, such as sequences of bit values. Figure 1.1 is an example of a neural network schema intended to learn associations between two data patterns, $pattern_1$ and $pattern_2$. We take $pattern_1$ and $pattern_2$ to be bit vectors, and the arrow from $pattern_1$ to $pattern_2$ in Figure 1.1 means that a directional association is being represented (e.g., learned from $pattern_1$ to $pattern_2$). By association learning we thus mean learning a correspondence from one data pattern ($pattern_1$) to another ($pattern_2$). If we want a network to associate $pattern_1$ with $pattern_2$, $pattern_1$ will serve as the input of the neural network, and $pattern_2$ as the desired output of the neural network. The task of the neural network is to learn to associate $pattern_1$'s with corresponding $pattern_2$'s.

While feedforward neural networks will be used to model association learning in this thesis, as they are relatively simple, other more complex methods have been

proposed. For example, Roy (2000; Roy & Pentland, 1998) modeled word learning as correlations across audio-visual associations. This was a computational model of word learning with multiple levels of acoustic and visual grounding (i.e., the model used various audio and visual input representations on multiple levels of learning). Input to this system was composed of color images paired with spoken utterances. Visual input was represented as color and shape histograms; audio input was represented as “phoneme traces” – probabilities of 40 phoneme classes and the most likely phoneme state sequence of the speech. These audio and visual representations are referred to as an AV-event. The model clusters AV-events for each possible combination of color and shape dimensions and applies a salience threshold. The remaining AV-events are used to build up an AV-lexicon. An “acoustic sequence spotter” computes co-occurrence statistics of spoken utterances in the AV-lexicon. The model includes short-term and long-term memory. The short-term memory has a limited capacity of approximately 20 words. The “recurrence filter” finds words that occur in similar visual contexts and computes acoustic metrics to find variations of the same phoneme sequences that account for the same words which are then stored in long-term memory together with the representation of the visual referent. Experiments with child directed speech showed that Roy’s (2000) model is able to extract 72% of the words and associate 57% of them with semantically correct referents (i.e., visual objects).

1.2.2 Category Learning Models

Categorization can be related to the human psychological concept of prototypes (Eysenck & Keane, 1995). In order for a data pattern to represent a member of a

category, the pattern should share some properties with other examples of the category. A prototype represents knowledge about the properties of patterns that are usually necessary for patterns to belong to a particular category. For instance, a prototype for a cup may be the knowledge that an object has a handle on its side, has a cavity pointing up and the fact that the bottom is flat. A prototype does not necessarily have to depict an example of object from a category. Category learning models typically are aimed to develop prototypes (i.e., representations of categories) for groups of similar inputs. In some models patterns are classified on the basis of prototypes. For example, Westermann's (2001) model forms prototypes to model categorical perception.

Categorization is often used to recognize contexts, e.g., presence of specific features at specific locations. Peterson (2001, 2002) proposes using a "context graph-based mapping architecture" for solving a navigation task. The architecture included three subsystems: context generation, a context association graph, and a motor selection subsystem. Context learning is accomplished with binding, storage, and context layers. The binding layer responds to specific combinations of input features at a specific location. The binding layer is connected to a storage layer that records a limited number of such salient bindings (cues), and constructs activity patterns. The context layer is implemented as an ART network (Carpenter & Grossberg, 1987) and responds to specific activity patterns by activating a node associated with a particular activity pattern (or allocates a node if none is associated with a particular activity pattern). Each node in the context layer is associated with a node in the context graph. Weights of the graph's pathways are updated whenever a context change occurs (e.g., an active node in the

context layer changes). The context graph is connected to a motor selection subsystem. The architecture also incorporates drive representations that influence context and motor command nodes using inhibitory and excitatory connections, and a context graph with motivational pathways. A motivational pathway acts as a reward when a performed action reaches its goal. Reported experiments with this context graph-based architecture embedded in a robotic agent included navigation in a maze and building a block tower. The system showed a robust real time performance and demonstrated the usefulness of context in problem solving.

Classification results depend on the features that describe the objects being classified. Weng and Hwang (2000) propose an Incrementally Hierarchical Discriminating Regression (IHDR) decision tree algorithm for classification and fast retrieval of high dimensional input. Image database approaches for feature selection (i.e., representation of images) include model-based and appearance-based techniques. A model-based approach entails manually selecting feature representations for a particular domain of images (e.g., face images). The approaches are limited to the application of the manually-selected features and impose a scaling-up problem for complex domains. An appearance-based approach entails selecting features from images represented as vectors based on statistical classification methods. Features derived in this manner may represent the images well but are not necessarily good for classification and vice versa.

An IHDR tree is built recursively. Each node of the IHDR tree has a fixed maximum number (q) of y -clusters with corresponding x -clusters. During training, the y part of an input pair (x_i, y_k) is used to find the closest y -cluster (using Euclidean distance)

and to update the corresponding x -cluster representation (mean vector and covariance matrix). The x -clusters are assumed to have Gaussian distribution. A y -cluster is represented by a mean vector and a diagonal covariance matrix. Examples are then assigned to a cluster by their y -value and x -cluster statistics are computed. Next, the samples are reassigned to clusters by the x -value based on the probability distance metric. If the Euclidean distance between the y_k 's in an x -cluster are greater than the sensitivity parameter δ_y , then a new node is created (in a similar manner). Size dependent negative-log-likelihood distance was used for assessing similarity of data points to a particular x -cluster. Weng and Hwang (2000) use a size dependent scatter matrix (SDSM) for negative-log-likelihood from Gaussian density. SDSM was defined as a weighted sum of three matrices: a covariance matrix, a within-class scatter matrix, and a scale matrix. The weights of the matrices are assigned as a function of the number of clusters (q), and the number of samples. High weights are given to the covariance matrix when large numbers of samples are available, and to the scale matrix when few samples are available. In a test, a label is assigned to the input x_i by finding the x -cluster with shortest probability-based distance from x_i .

The performance of the IHDR algorithm was compared with CART, C5.0 and OC1, including preprocessing of data with PCA (principal components analysis) on appearance-based face image retrieval tasks. IHDR was able to perform without error, while all other algorithms had error of 41% or more. Also IHDR showed a good and robust performance in a navigation task.

In the present thesis we assume that some category learning model was used to categorize the data and that the category information is available for use. We then will use this category information in association learning models to assess its effects on association learning.

1.2.3 Interference Reduction Techniques

Interference is said to occur when learning a new task changes the performance of a system on a previously learned task. The interference effect is often observed in sequential learning tasks. In a sequential learning task a neural network is trained on examples from category A, and then it is trained on examples from category B. A typical outcome of a sequential learning task for feedforward multilayer neural networks is large – and in some cases complete – loss of performance on examples from the initial training set (i.e., category A examples). This is called catastrophic interference. In contrast to a sequential learning task, a typical neural network learning task comprises training on examples from multiple categories simultaneously. In a simultaneous learning task, all patterns are available at the same time for training. Therefore, neural networks trained in a simultaneous fashion may have similar performance for patterns from all categories. That is, interference does not typically occur in neural networks with simultaneous learning. Many association learning tasks are learned by humans in a sequential manner, since examples from only a few categories may be available at a given time (French, 1999). Interference may also occur in human learning, but in contrast to neural networks, gradual interference may be observed in human learning. We hypothesize that providing category information to association learning will reduce interference to gradual rather

than catastrophic levels. This particular benefit is of most interest to us, because it will make neural networks more closely model human memory.

Ratcliff (1990) explores different strategies for reducing interference in neural network learning including selective weight updates, rehearsals, and context models. Ratcliff examines models of learning and forgetting in an encoder task: given an input, a network should reproduce that input as the output using a relatively small hidden layer of units ($\frac{1}{2}$ or $\frac{3}{4}$ the number of units in the input layer). The first simulation with the encoder network used a four-bit binary input vector, of which only one bit could be active (i.e., set to 1) in a vector. The network was trained on a first training set consisting of three vectors; and then trained on a second training set of one vector using one of the following strategies: 1). Modify all weights; 2). Modify weights that were not significantly changed after initialization; 3). Add new hidden units and modify all weights; 4). Add new hidden units and modify only weights associated with the newly added hidden units. The result of these strategies was essentially that the network was not able to reproduce items from the first training set after learning items from the second training set. That is, the network showed catastrophic interference. Simulations with more than one item in the second training set and similar simulations with larger input vectors (32 bits) yielded essentially similar results with the above strategies still showing catastrophic interference in association learning.

Another approach explored by Ratcliff (1990) was the *rehearsal buffer model*. This model uses a buffer that contains either one or the four last items and each item is trained for N epochs with other items in the buffer before moving to training of another

item. This model is based on the idea that “in list learning [e.g., a sequential learning task], groups of items are rehearsed together” (Atkinson & Shiffrin, 1968; Rundus & Atkinson, 1970; p. 293; as cited in Ratcliff, 1990), and results of previous models described above that showed somewhat better recall when the training sets contained multiple items. Experiments included simulations with 32, 128 and 1024 bit input vectors and the previously used encoder task. The main finding of the experiments is that learning new items destroys knowledge of previously trained items, i.e., they showed catastrophic interference. Analysis of the network’s output on the test sets showed that the output took on one of three values: 1) the last learned item, 2) an average of the training patterns or 3) the most similar training pattern depending on the size of the buffer.

Ratcliff (1990) also explored using context as part of the input in order to overcome catastrophic interference in neural networks. He defines learning as the ability to discriminate between previously learned and new items as a function of training time in the context of a sequential learning task. In the training phase, a network learns to produce a context vector for a particular set of items. In a testing mode, the network should then produce a different context vector for new items, and match the context vector for previously learned items. The result was that a network would have a different context vector (from the learned context vector) for both old and new items in a test. Variations of this model included sequential learning task simulations on networks that had received some initial training (initial training was not on patterns from the sequential learning task). This was inspired by the fact that humans in many cases already have

some knowledge prior to training on a sequential learning task. Sequential learning task simulations with such initially trained networks included data sets both with and without context information. The models did not provide evidence of reducing interference as a function of training duration.

French (1999) reviews possible causes, methods for measuring, and possible ways of overcoming catastrophic interference in multilayer feedforward neural networks. In many cases, the effect of catastrophic interference in a sequential learning task is due to neural networks having only a single set of weights that are changed during training on the second training set. During training on the second set of examples the weights acquired by training on the first training set are overridden. In contrast, humans in some cases have gradual interference. That is, in a similar sequential learning task with humans, examples from the first set may be forgotten, but only to some degree rather than completely.

Two methods are commonly used for measuring interference. In the exact recognition method, after training on a second set of examples the recall of examples from the first training set is measured. Another measure of interference (Hetherington & Seidenberg, 1989; p. 131; as cited in French, 1999) includes assessing the time required to restore a neural network to the state where it can recall examples from the first training set after having been trained on a second training set. This method considers the case that after learning the second training set a network may be able to quickly relearn associations from the first training set.

Methods for overcoming catastrophic interference in neural networks, as described by French (1999), include reducing the overlap of input pattern representations, and also internal representational overlap (e.g., overlap in hidden units activations). Reducing input representation overlap may be achieved by using sparse input vectors that will lead to only a few hidden units being active for a particular input, which also reduces hidden representation overlap. This technique may be natural when input units are from highly structured domains such as language or fact learning, and therefore are themselves highly structured. Another way of reducing input representation overlap is using bi-polar feature coding (i.e., -1 and 1) instead of the more common codings (e.g., 0 and 1). Drawbacks of reducing representational overlap are lower generalization performance and reduced ability to discriminate between new and known patterns. Also this method seems to be unable to reduce interference for large numbers of patterns. Attempts at reducing internal representation overlap by orthogonalizing hidden layer activations patterns leads to somewhat local representation (i.e., not distributed over all network's weights, or semi-distributed weights) of an input in the hidden layer. An example of a neural network that uses this technique is an ART network (Carpenter & Grossberg, 1987) that groups similar input patterns by making a local internal representation for them. It is local in a sense that when an ART network forms a representation for a new group of input patterns, the weights representing other groups are not significantly disrupted. Finally, interference may be reduced in neural networks by using dual network architectures for learning. In this approach one of the networks learns new examples and the other accumulates knowledge from all training examples by learning from the first

neural network. These architectures parallel, to some degree, the way that the brains of various animal species are wired. In these animals, one brain region (the hippocampus) learns new information and then forwards its knowledge to another region (the neocortex) that is responsible for long-term accumulation of knowledge (French, 1999). Neural networks with dual-net architecture also may include variations with pseudo-rehearsal mechanisms for transferring knowledge from one network to another.

Ans and Rousset (2000) combine the ideas of rehearsal and using multiple neural networks for sequential learning without catastrophic interference. The proposed architecture consists of two coupled neural networks, NET1 and NET2 (see Figure 1.2, page 19). Both networks use a feedforward architecture with one layer of hidden units with the following modification: hidden units are fully connected to the input layer. The error function used in this architecture is a modified backpropagation error function. The error function considers both errors associated with the target output and errors generated by the hidden units' activations to the input units when learning an external input. The external input comprises associations from training sets. Internal input comprises pseudo-associations (i.e., input-output pattern pairs generated using the other network). External input is learned by NET1, and NET2 stores previous knowledge for use in rehearsal. After initial training of NET1 to the criterion on external input, further training is accomplished in two stages. In stage 1, information from NET1 is transported into NET2 by generating pseudo-associations and training NET2 using these pseudo-associations. Thus NET2 combines knowledge from past tasks with the knowledge from the current task. In stage 2, NET1 learns associations from a training set to a learning criteria

simultaneously with pseudo-associations generated by NET2 (pseudo-associations are not considered when evaluating the training criteria). Generating a pseudo-association is done in the following steps. A random noise generator creates input patterns that produce some activation in the hidden layer units. This activation is reinjected to the input layer to generate input-output pattern pseudo-association. After a specified number of reinjections from the hidden layer to the input layer (e.g., reverberations) the input-output pseudo-association is passed on for the other network's training.

The Ans and Rousset (2000) simulations with reverberating networks with a self-refreshing memory model includes sequential and simultaneous (i.e., mixture of input patterns from all categories) presentation of learning tasks. They ran simulations with both compatible (decimal and octal addition) and incompatible (numeric maximum and octal addition) tasks. Input patterns consisted of two two-digit numbers, each digit represented with three bits (no eight's or nine's were used). Input patterns also included two-bit operation encoding. Output was a two-digit number represented by six bits (three bits for each number). The operand numbers were in the interval from 1 to 47 and their octal sum had at most two digits (i.e., less than 64). The operation encoding in the input was implicit category information. From a more general view, those two bits can be considered as category information representing the type of information in an association learning task. The main result is that the proposed architecture is able to overcome catastrophic interference in neural networks. Experiments showed that self-refreshing significantly improved recall performance for both compatible and incompatible learning tasks (recall reached 100% correct performance in both cases) when presented in a

sequential manner. Learning both compatible and incompatible tasks showed somewhat lower generalization performance, and somewhat longer training times for sequential as compared to simultaneous tasks. Generalization performance for the sequential compatible tasks was approximately 90% compared to approximately 70% for concurrent presentation of the compatible tasks. It took approximately 325 training epochs to reach the training criterion (0.1 per sample error) when the tasks were presented sequentially compared to approximately 550 training epochs when the tasks were presented concurrently. Generalization performance for the sequential incompatible tasks was approximately 50% compared to approximately 55% for the concurrent presentation of incompatible tasks. It took approximately 1400 training epochs to reach the training criterion when the tasks were presented sequentially compared to approximately 850 training epochs when the tasks were presented concurrently. Both sequential and concurrent presentation of compatible tasks (decimal addition, then octal addition) showed improvement in recall, generalization performance and training duration over learning a single task (octal addition). In conclusion, a reverberating network with a self-refreshing memory model is able to overcome catastrophic interference, and thereby more closely models interference effects in humans.

1.3 Overview of Experiments

We now turn to the simulations of the present thesis. As we have stated, our goal in this thesis was to explore how category information affects association learning in computational models of neural networks.

Experiment 1 was designed to assess the effects of category information on association learning as measured by learning rate and generalization performance. This experiment compared two models of association learning. Model₁ learned associations between pairs of patterns, while Model₂ learned the same associations but combined category information with pattern₁. Experiment 1 utilized a simultaneous learning task.

In Experiment 2A we further investigated the affects of category information on association learning. This experiment used more complex associations than those in Experiment 1 which utilized associations that were relatively easy for a neural network to learn. Experiment 2B used the same data as Experiment 2A and the same models of association learning (Model₁ and Model₂), but used a sequential learning task instead of a simultaneous learning task.

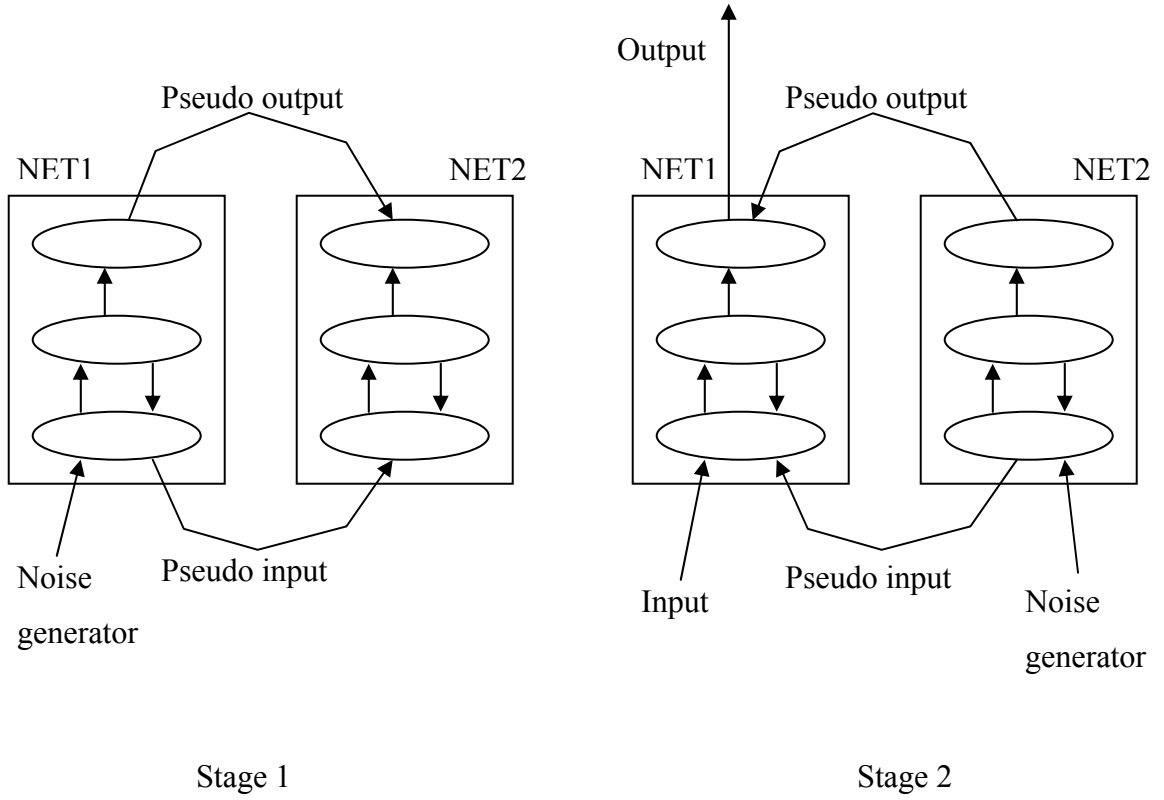


Figure 1.2 Revebrating Architecture with Self Refreshing Memory (From Ans & Rousset, 2000)

Chapter 2. Experiment 1

2.1 Introduction

The overall goal of this research is computational methods of learning associations. We are using categorization techniques in order to potentially improve methods of association learning. In this experiment our goal is to test the hypothesis: Does providing additional, precategorized, input affect the learning rate or the generalization ability of association learning?

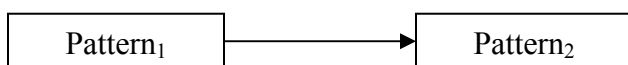
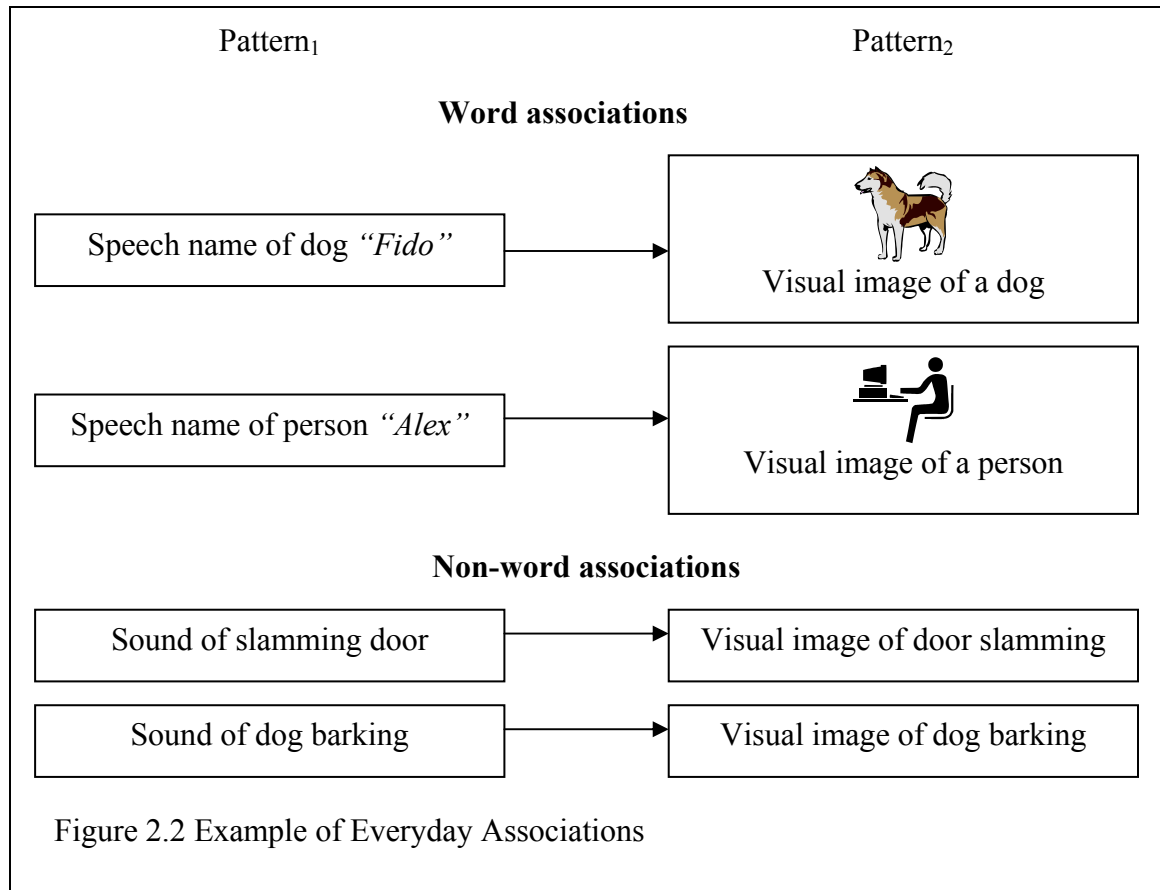


Figure 2.1 Model₁ Schema of Association Learning

By association learning, we mean establishing some relationship between pairs of data patterns. These pairs of patterns represent data from the real world in a form that is suitable for processing by a neural network. We use neural networks to carry out association learning. Figure 2.1 is an example of a neural network schema intended to learn associations between two data patterns, pattern₁ and pattern₂. Pattern₁ and pattern₂ can be, for example, sequences of bit values. By association learning we thus mean learning a correspondence from one data pattern (pattern₁) to another (pattern₂). We will refer to the association learning between pattern₁-pattern₂ pairs as Model₁.

We are interested to see if the addition of category information into the association learning can alter the association learning. For example, we might

pre-categorize the input, or add category information to the data in the patterns. We hypothesize that this may improve the association learning (e.g., increase the learning rate or generalization performance).



By way of motivating interest in this problem, it is easy to see that humans learn associations. For example, we associate the sight of a person with the sound of voice. An association learning example in an everyday context is presented in Figure 2.2. Each instance of $pattern_1$ is associated with a particular instance of $pattern_2$ in the examples in Figure 2.2. We can use a $Model_1$ schema to represent the learning of associations between sounds and sights if $pattern_1$ represents a sound and $pattern_2$ represents a visual image. Associations in Figure 2.2 can be categorized as word/non-word associations

based on whether pattern₁ is a speech or non-speech sound. It may be the case that categorical information can assist humans in learning associations. For example, if we expect to hear a word, this may enable us to apply word learning strategies.

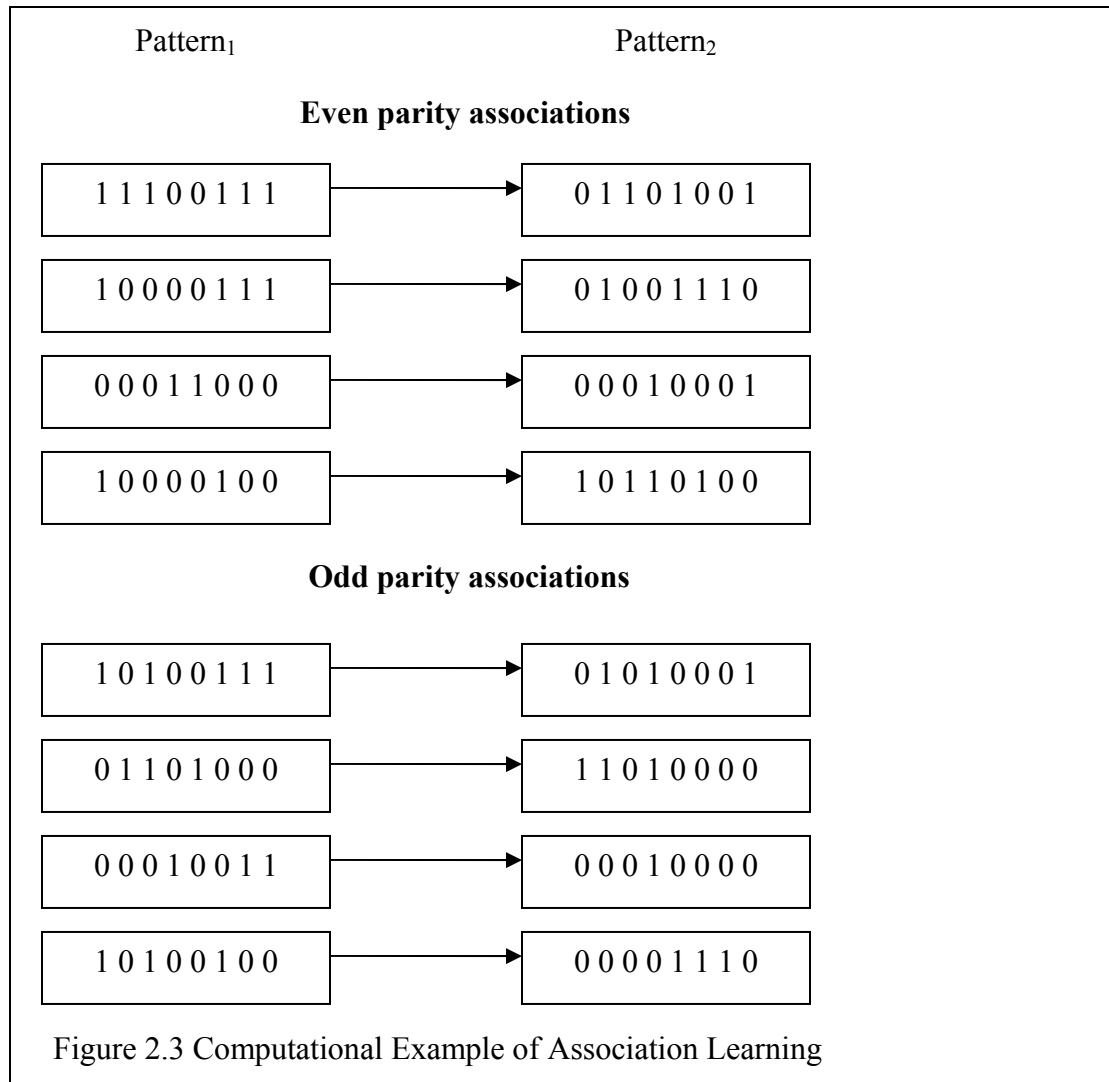


Figure 2.3 is a more computational example of association learning involving categories across different associations. Each pattern has eight bits. Association categories are formed on the basis of parity. When both pattern₁ and pattern₂ have an

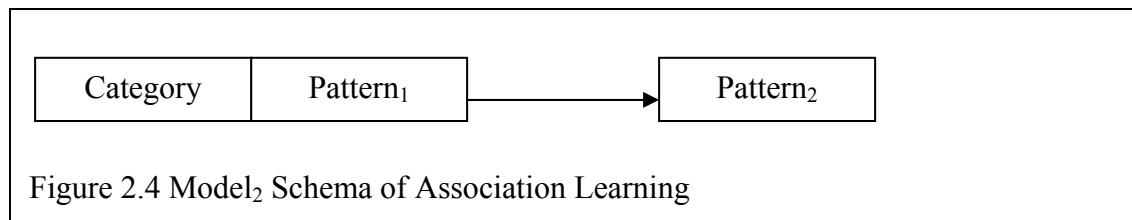
even number of 1 bits, they form an even parity association. When they have an odd number of 1's, they form an odd parity association.

The examples in Figure 2.2 and Figure 2.3 introduce the idea that associations can sometimes be divided into categories. In Figure 2.2, there were word associations on the one hand and non-word associations on the other hand. In Figure 2.3, there were even and odd parity associations. We hypothesize, that if we can incorporate such category information into a neural network method of association learning, then we may be able to improve the association learning.

In formulating this hypothesis, we are making an analogy with human learning: sometimes humans learn better when they know the category of what they are learning. For example, when you look at a book, by some properties of the visual image (e.g., properties as shape, size, material) you automatically categorize the object as a book and start looking for properties of books (e.g., the title or author). This illustrates the human ability to very quickly categorize the visual input as a particular type of object, and based on this, explore the visual input for some specific features (such as a book title).

In both everyday associations and more computational examples, a network may learn associations between $pattern_1$ - $pattern_2$ pairs. But as we have seen the $pattern_1$ - $pattern_2$ pairs themselves may sometimes be grouped into categories (e.g., Figures 2.2, 2.3). In the everyday associations example (i.e., Figure 2.2), pairs could be grouped with respect to word/non-word association based on whether we have a speech or non-speech sound and in the computational associations example, pairs could be grouped with respect to even or odd parity. We may be able to utilize the pattern's structure to pre-

classify patterns into categories and provide this category input along with the data input. In Figure 2.3, the first three associations can be preclassified as Category₁, as the number of 1s is even, and the last three associations can be classified as Category₂ (odd number of 1s). With the use of categorization information in mind, we modify the Model₁ to include precategorized information. We will refer to the schema with category information as Model₂ (see Figure 2.4).



We assume that category information can be represented as a sequence of bits. Figure 2.4 is an example of association learning using a Model₂ schema. Category information is presented as one additional bit of the pattern₁. Category₁, even parity association, is represented by category = 0. Category₂, odd parity association, is represented by category = 1. Associations would now be learned between (category, pattern₁) and pattern₂ pairs.

Experiment 1 is designed to test the following hypotheses with Model₁ and Model₂ instantiated as neural networks:

- 1). When Model₁ learns associations, the performance will be approximately the same for pattern₁, pattern₂ pairs of the two categories.
- 2). When Model₂ learns associations, the performance will be approximately the same for pattern₁, pattern₂ pairs of the two categories.

3). Model₂ will show general improvement in performance over Model₁ in regard to association learning.

Performance measures will include the number of trials of training and generalization tests.

2.2 Methods

To test these hypotheses, we have implemented Model₁ and Model₂ as neural networks. Association learning as we have defined it implies that both patterns are known, and the goal is to establish relationships between pairs of patterns. Therefore, a supervised neural network training method can be used for association learning. In this experiment we used backpropagation (Rumelhart, Hinton, & Williams, 1986) for training associations between pattern₁ and pattern₂. For Model₁ neural networks, the input pattern corresponds to pattern₁ and the desired output corresponds to pattern₂. For Model₂ neural networks, the combination of category and pattern₁ correspond to the network's input, and the desired output corresponds to pattern₂.

2.2.1 Neural Network Architecture

We have used fully-connected feedforward neural networks with two layers of weights (i.e., one layer of hidden units) in this experiment. A schematic view of the network architecture is presented in Figure 2.6 (page 36). The Figure 2.6 neural network with zero category units corresponds to Model₁. The input layer has eleven units which correspond to the eleven bits of pattern₁. The output layer has two units, these

corresponding to the two-bit pattern₂. We have used eleven bits for pattern₁ in order to have a sufficient variety of training and testing patterns.

Figure 2.6 with at least one category input unit (i.e., at least one category bit) corresponds to Model₂. The number of input units for Model₂ corresponds to the combined dimensionality of the pattern₁ and category data vectors.

We have used a constant learning rate of 0.45. In the training phase, the criterion for termination was that all training data patterns had a per sample error no greater than 10%. Per sample error, ε_s , was computed as follows. For the actual output a , desired output d , and worst case output w , Euclidean distances e_w and e_a were calculated (Equations 2.1, 2.2). e_w represents the greatest (100%) possible per sample error and e_a gives the actual error. ε_s was computed as shown in Equation 2.3

$$e_w = \text{EuclideanDistance}(d,w) \quad (2.1)$$

$$e_a = \text{EuclideanDistance}(d,a) \quad (2.2)$$

$$\varepsilon_s = \frac{e_a}{e_w} \times 100\% \quad (2.3)$$

The maximum number of training epochs for networks in the experiment was 20,000. In the generalization, a particular test was considered correct if the relation between the output units was in the desired or expected direction. For example, for actual activations of output units a_1 and a_2 , if $a_1 > a_2$, this was considered correct only when desired output units' activations were $d_1 > d_2$. We used a batch training mode because in preliminary evaluation incremental training was slower. All neural network parameters

except the number of input units (i.e., the presence of category units) were the same for Model₁ and Model₂ networks.

2.2.2 Data Sets

In some cases categorization can be considered as a type of association learning. Learning that a set of exemplars is of particular category can also be considered as learning that each of the exemplars is associated with a category. In the present experiment, the network learns to associate exemplars with one of two category values. The association being learned was labeling input patterns with a particular output based on the parity rule. The pattern₁ data vectors were categorized as category₁ (even parity) or category₂ (odd parity). When pattern₁ was from category₁, pattern₂ was 0 1. When the pattern₁ was from category₂, pattern₂ was 1 0. Category input units, where applicable, were set to all 1 for category₁ and to all 0 for category₂.

We assessed the effects of two variables on the performance of the association learning: the number of associations, and the number of category units in the network. We used 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 unique input patterns for training a network, and 250 unique input patterns to test generalization. Half of the training patterns were from category₁, and the other half from category₂. Also, half of the test input patterns were from category₁, and the other half from category₂. To ensure a balanced data set design, data sets with 32 or more input patterns were generated using blocks of 32 patterns, where a block contained 16 category₁ samples and 16 category₂ samples, and within the block the samples were randomly ordered. A total of 100 simulation runs were used for every combination of input patterns and category units, and in each simulation

run a network was initially assigned randomly different weights. Five data sets were used for each combination of input patterns and category units. Due to training epochs consistently exceeding the 20,000 epoch limit, only ten simulation runs were used for zero category unit architectures with 64, 128, 256 and 1024 input patterns.

2.2.3 Performance Measures

Two performance measures were used to assess association learning: duration of training and generalization. Duration of training was measured as the number of training epochs until the performance criterion or 20,000 epochs was reached. We assessed generalization by the ability of a network to associate new patterns with the correct output pattern. We used the ratio of the number of correct test associations to all test associations as a measure of generalization. We used mean number of trials and mean generalization ratios across the 100 (or ten) networks simulated.

2.3 Results

Figure 2.7 (page 37) summarizes the results of training and gives the number of training epochs. Data are presented as a function of the number of input patterns, and the number of category units in the network architecture. The main result of training was that the network with zero category units tended to take far longer to learn the associations. For example, with 16 training patterns, the zero category unit architecture took on average 238.23 training epochs to learn the associations while all architectures with at least one category unit took no more than on average 68.55 epochs for the same number

of training patterns. Figure 2.8 (page 38) shows training epochs for the zero category units architecture.

Other notable features of Figure 2.7 include little difference between the performance of architectures with single and multiple category units, and a general trend to fewer required training epochs for these architectures with larger training sets. The downward trend in the number of training epochs may be explained by the variable number of training samples per training set. The actual number of training patterns processed by a network increased with the size of training set. For example, for the one category unit architecture $153.18 \text{ epochs} \times 2 \text{ training patterns}$ is approximately 306 patterns processed in training by each network, and $16.63 \text{ epochs} \times 1024 \text{ training patterns}$ is approximately 17,029 patterns processed by each network. If the training data were to be scaled by the number of training patterns, the downward trend in number of training epochs required would change to an upward trend in number of processed patterns. The findings of little difference in performance of networks with at least one category unit is likely due to the fact that the required output (0 1 or 1 0) was a simple function of the category unit input, for example, with category input of 1, the required output was 0 1.

Figure 2.9 (page 39) shows generalization performance averaged across the two categories. Networks with zero category units generalized correctly on at most 62.4% of tests while networks with categorical input reached 100% accuracy in generalization tests in a majority of cases. In order to assess if the performance of the zero category units architecture was different than that expected by chance we computed a cumulative binomial statistic for each data point. Table 2.1 (page 42) presents the cumulative

probability of getting the zero category units architecture's correct number of outcomes by chance (i.e., randomly assigning category outputs). The probability has a binomial distribution with n equal to the number of trials, and chance probability of successful outcome, p , equal to 0.5. Cumulative probabilities of getting respective number of successful outcomes for training sets with 2, 16, 512 and 1024 training patterns has $p < 0.00008$ which suggests that the zero category units architecture is able to generalize from associations learned during training. The generalizations it made do not appear to be selected at random. The probabilities for data sets with 2 and 16 training patterns are not reliable. Additional simulation runs (on different data sets) showed that the average percent of correct tests in these cases varied from 49% to almost 52% on a single data set. The probabilities for data sets with 512 and 1024 training patterns are likely due to the large number of training patterns processed by each network enabling better generalization.

Figure 2.9 (page 39) also shows the variability of accuracy in the generalization tests for networks with category inputs for relatively small amounts of training data – less than 32 input patterns. Such results with small training data sets might be explained by the size of training data, as for instance, one or two input patterns do not represent category₁ adequately, i.e., this is just not enough training examples for making correct generalizations.

Figures 2.10 (page 40) and 2.11 (page 41) illustrate generalization results with respect to each category. Figure 2.10 shows that networks with no categorical input have a relatively high degree of variation in generalization test results for the two categories.

The variation for small data sets (64 and less training patterns) is likely due to the small number of training patterns. A small data set does not provide a representative sampling of the problem space. Data sets with 128 and 256 training patterns had a relatively small variation in generalization test results for the two categories (less than 2%). Although the networks' training was not completed within the 20,000 training epochs limit for these two data sets, the networks were very close to the completion of training (no more than 5% of training patterns had per sample error over 10%). The high variation of generalization test results for large data sets (512 and 1024 training patterns) is likely due to larger number of training patterns with per sample error (over 13%). We ran additional simulations with 200,000 maximum training epochs limit and 0.1 learning rate to assess the variation in generalization test results of the Model₁. Results of generalization performance tests are presented in Figure 2.12 (page 43). We found that varying the limit of training epochs and learning rate did not significantly affect generalization for data sets with 128 and less training patterns. Both categories had approximately the same generalization performance (less than 5.52% difference) for data sets with 128, 256, 512 and 1024 training patterns. Also there is an upward trend in generalization performance results for these data sets. These results are likely due to large number of training patterns and more complete training (number of training patterns with per sample error was less than 6%) that enabled better generalization.

Figure 2.11 gives results for the one category unit architecture, and is shown as a typical example of generalization tests for architectures with category input units. Tests yield similar performance in generalization for both categories with 32 and more training

patterns for architectures with 1-4 category units (typically is 100% correct). Figure 2.11 shows that architectures with one category unit have variable performance on generalization for the two categories with the size of training data less than 32 patterns. This is typical for all architecture with category units. Such results with small training data sets may be due to the small size of training data set. A small data set does not provide a representative sampling of the problem space to enable correct generalizations.

2.4 Discussion

In this experiment we assessed the influence of the number of training patterns and the number of category units on the performance of association learning in terms of learning rate and generalization. Results of the experiment showed that category units in the input always reduce the number of training epochs and increase the level of generalization of networks relative to networks with no category units.

Considering our hypotheses, based on the results of this experiment we can state that:

- 1). Different than our expectations, Model₁ did not always have approximately the same performance for the associations from category₁ and category₂ (see also Figure 2.10, page 40). Models with and without category input units have variation in generalization performance with few input patterns, but Model₁ showed variation across most of the data sets. Such results with small training data sets may be due specifically to the size of training data sets. For instance, one or two input patterns do not represent category₁ adequately. These are just not enough training examples to enable correct generalizations. Model₁ did not reach the training criteria (10% per sample error) within

the 20,000 training epochs limit with more than 64 training patterns. This is likely the reason for variation in generalization performance with large data sets, as with incomplete training there was quite a large percent (20.6% for the 1024 training patterns data set) of training patterns with per sample error. Additional simulation runs of Model₁ with 200,000 training epochs limit and 0.1 learning rate showed similar generalization results with less than 256 training patterns data sets, and that with large data sets (256 and more training patterns) Model₁ had an upward trend in generalization, reaching 75.2% accuracy and having approximately the same generalization performance for associations from both categories. In order to achieve similar performance for associations from both categories with relatively high generalization performance (over 65% of correct tests), Model₁ needs larger amounts of training with comparatively large data sets.

2). As expected, Model₂ had approximately the same performance for associations from both categories when the training data had more than 32 training patterns, reaching 100% accuracy in generalization in most cases (see also Figure 2.11, page 41). Such accuracy is likely due to the fact that the required output (0 1 or 1 0) was a simple function of the category unit inputs. The variability of the Model₂ performance in generalization with less than 32 training patterns may be explained by the small size of training data set that does not adequately present the problem space.

3). As expected, Model₂ showed improvement in performance over Model₁ in regard to association learning. Model₂ had higher performance in generalization than Model₁, as well as faster learning rates. The improvement in the Model₂ performance is due to adding category units to pattern₁'s.

In these simulations we experimented with categorical associations. More specifically, in this experiment $pattern_1$ was eleven bits, $pattern_2$ had one of the two possible values (01 or 10), and instances of $pattern_1$ were associated with instances of $pattern_2$. Clearly, associations can be more complex. For example, associations learned between visual and audio data can be much more complex. Consider the problem of learning an association between an image of a dog and the word “dog” (see also the upper most association in Figure 2.2, page 21). Assuming input of both color and shape features in the visual data, associating the visual information *yellow dog* with the sound “dog” may be far more complex than the present stimuli. In the next experiment we investigate the affect of pre-categorized input on learning rate and generalization ability of association learning in a case with more complex associations.

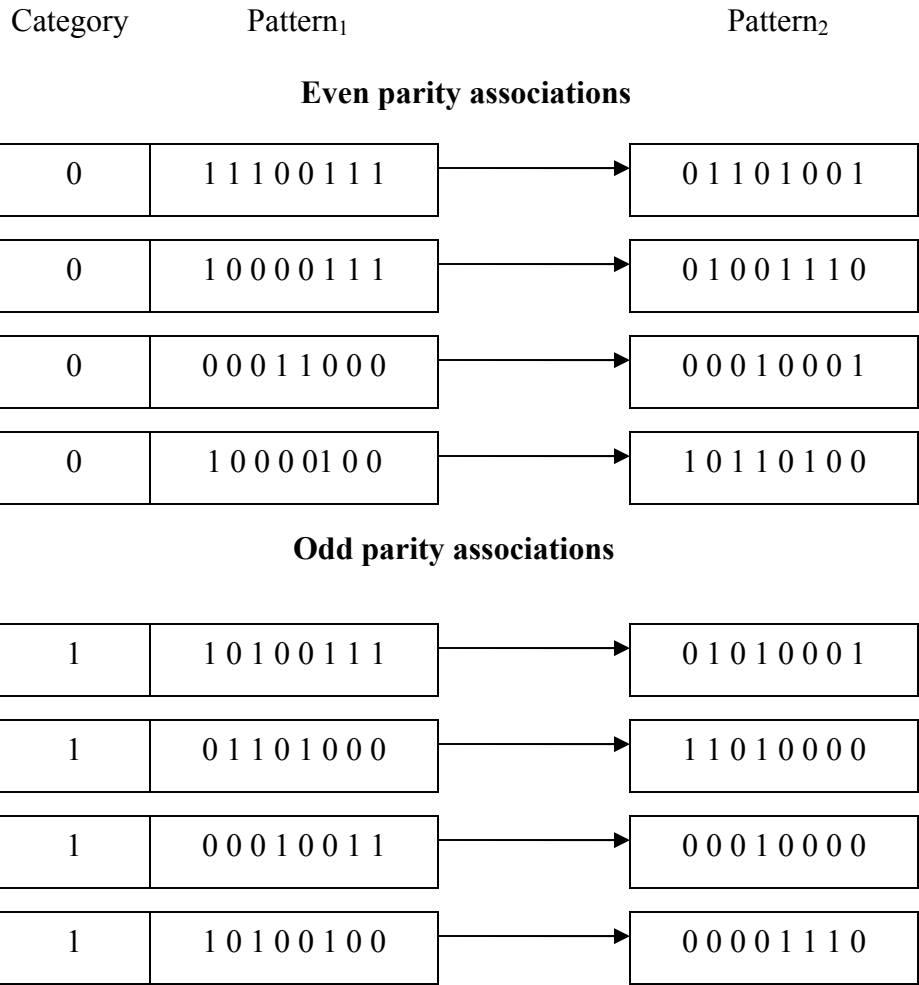


Figure 2.5 Computational Example of Association Learning With Category Information

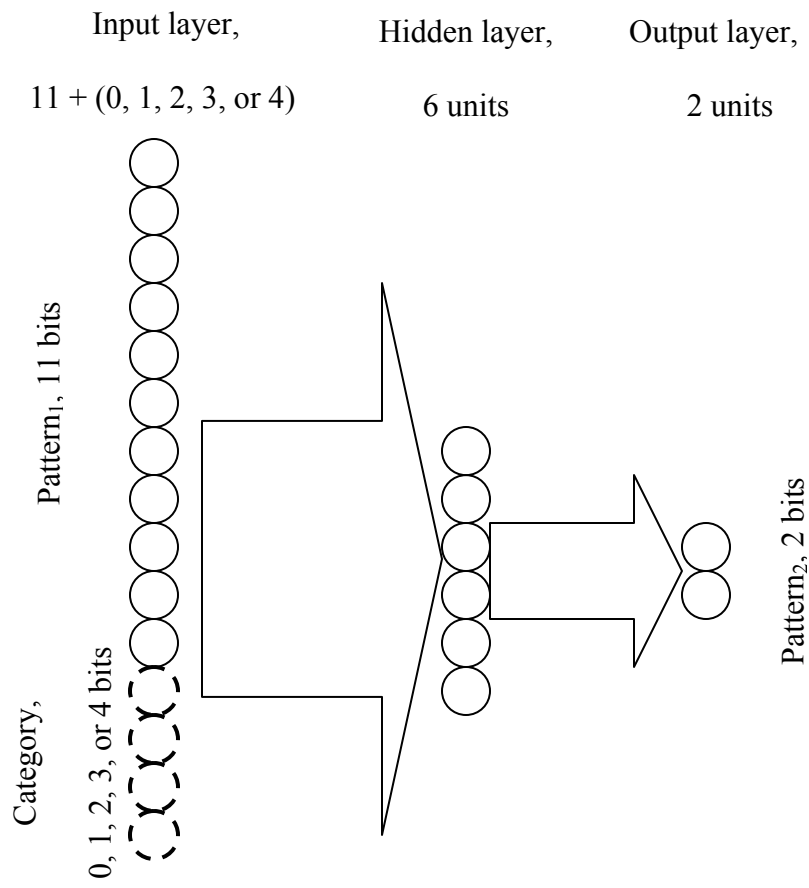


Figure 2.6 Network Architecture

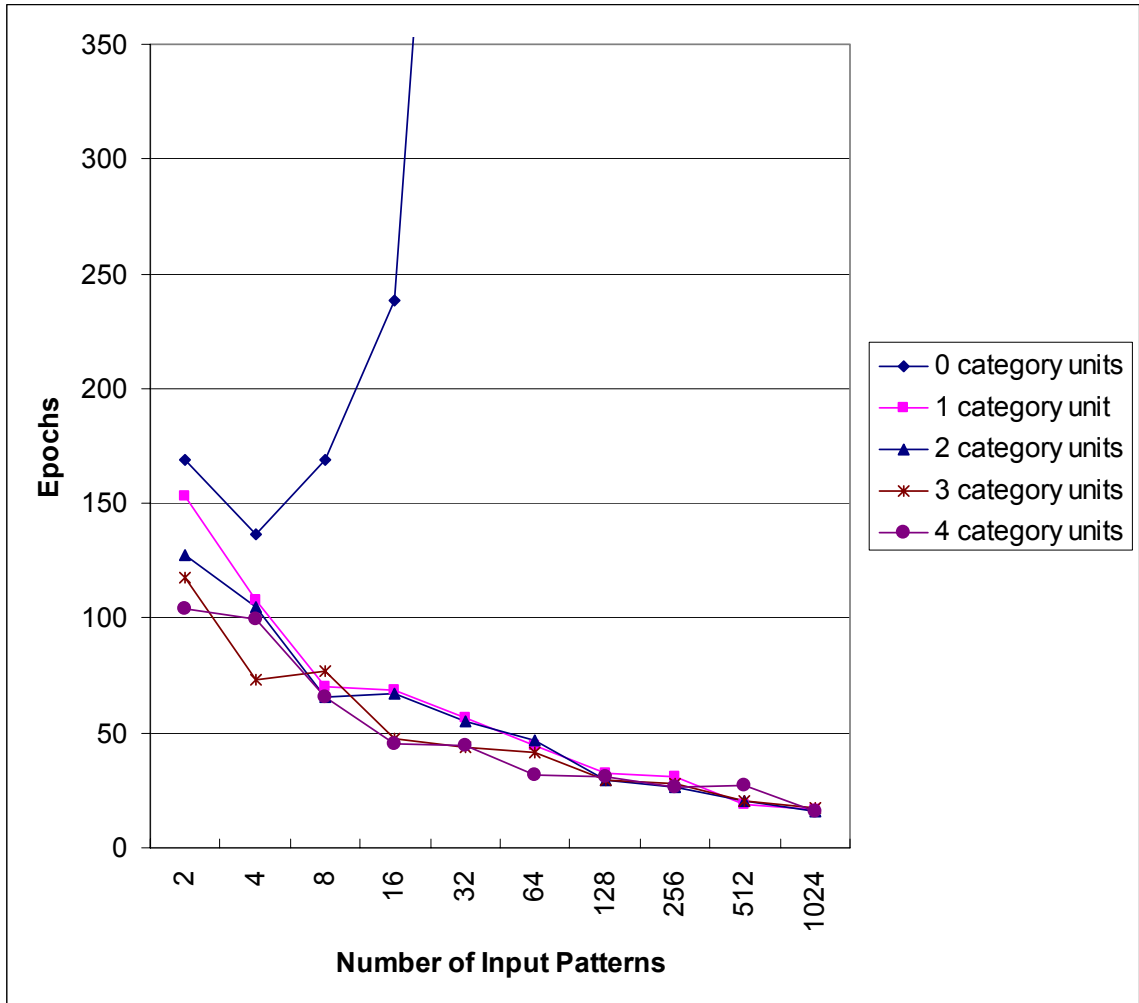


Figure 2.7 Amount of Training Required for All Architectures

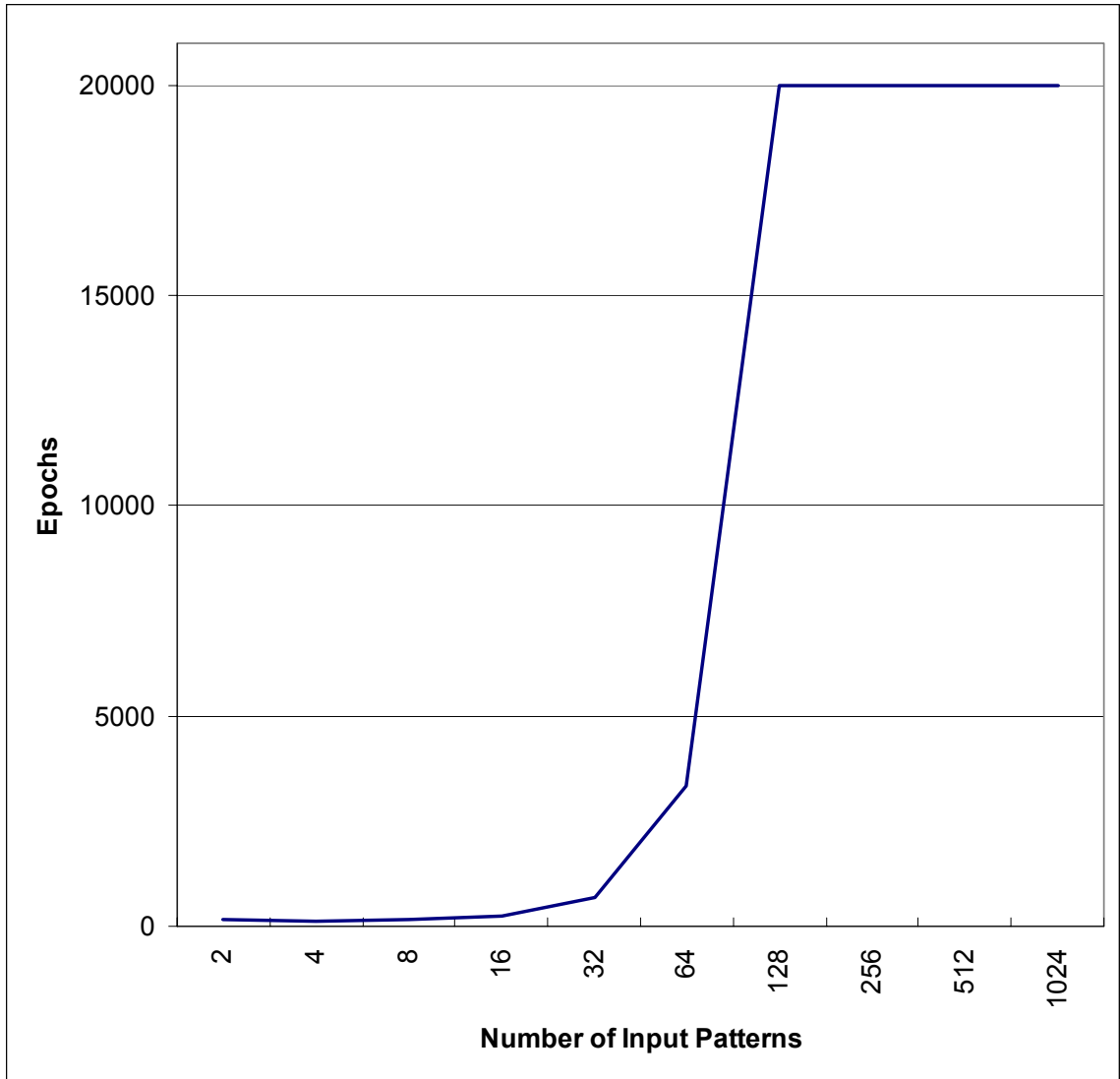


Figure 2.8 Amount of Training Required for Zero Category Units Architecture

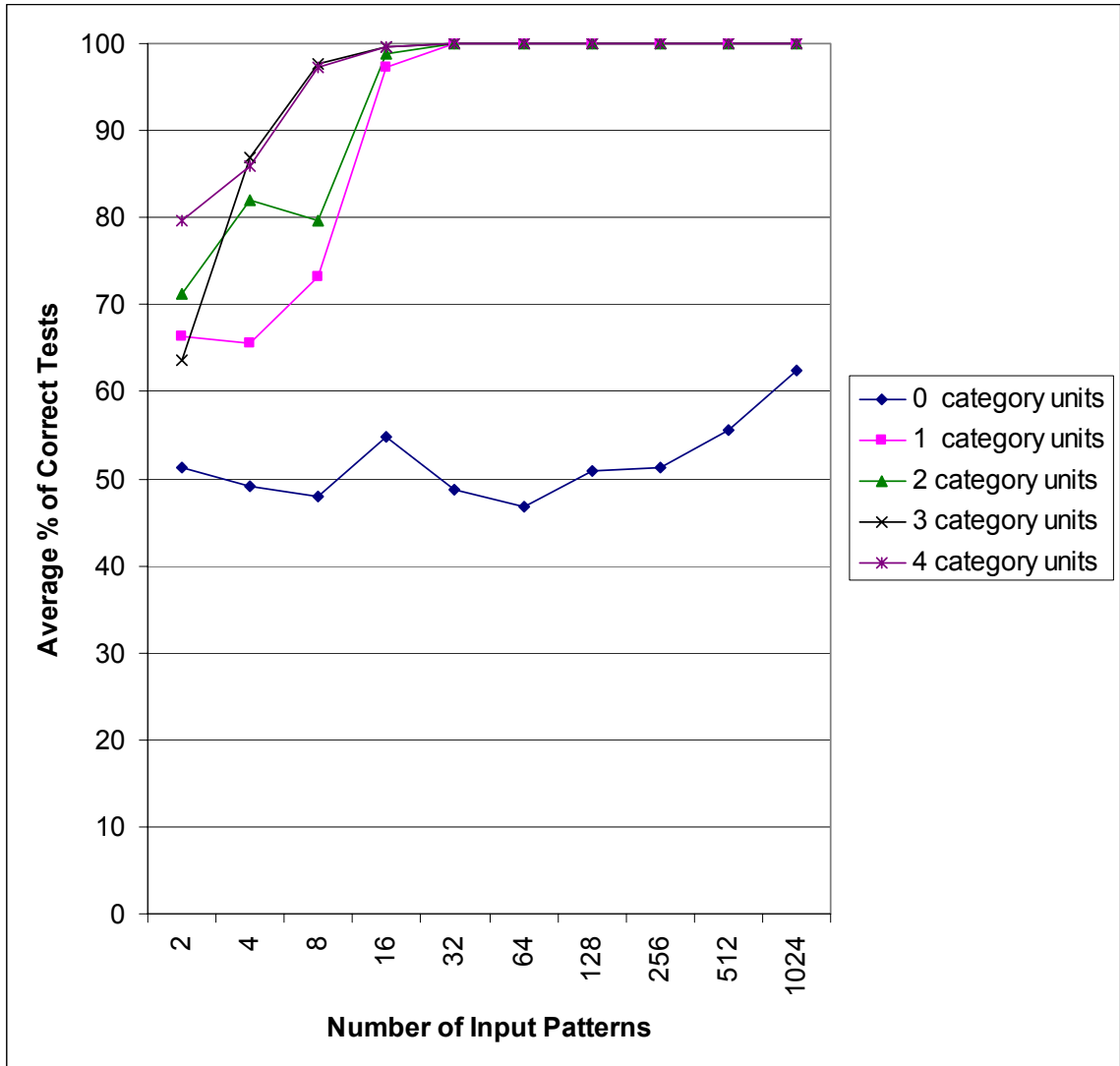


Figure 2.9 Generalization Performance for All Architectures

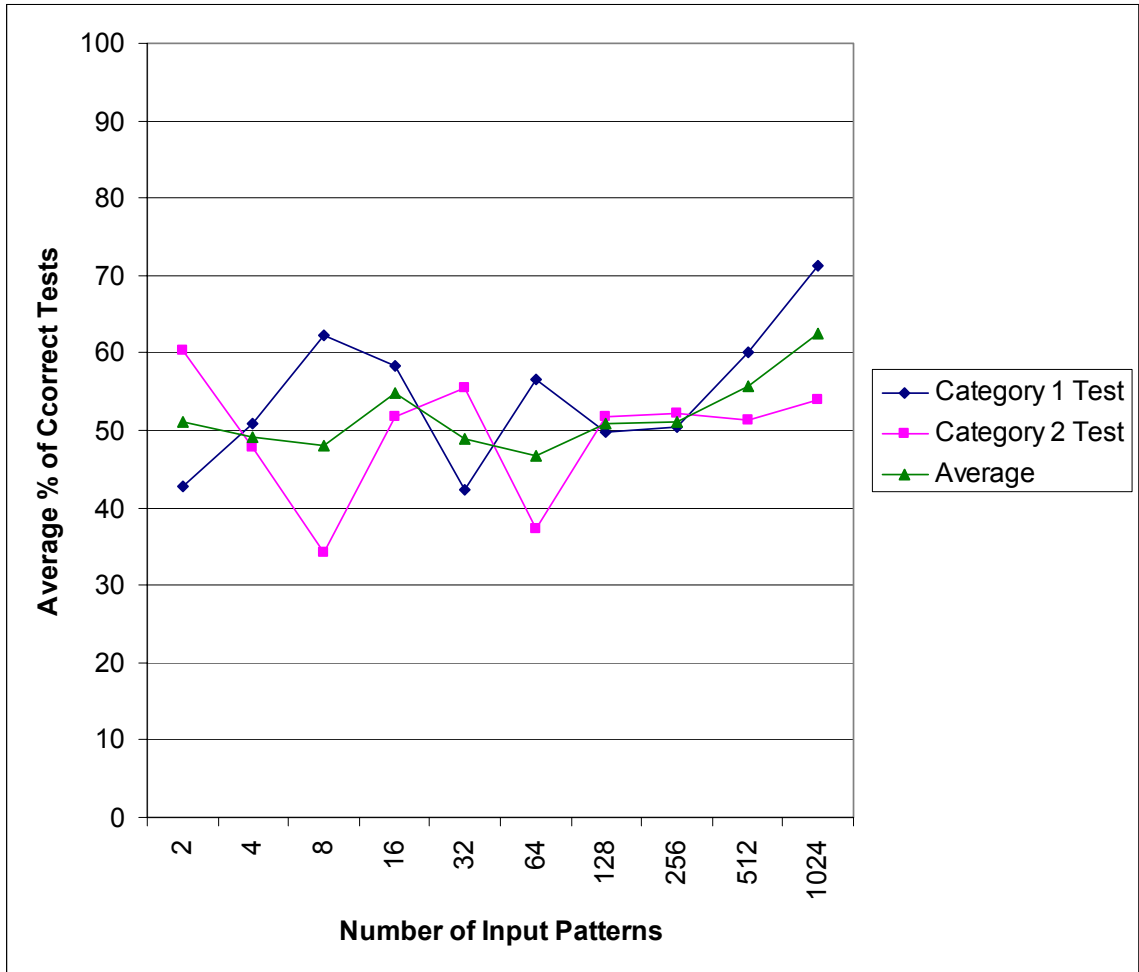


Figure 2.10 Generalization Performance by Category for Zero Category Units Architecture

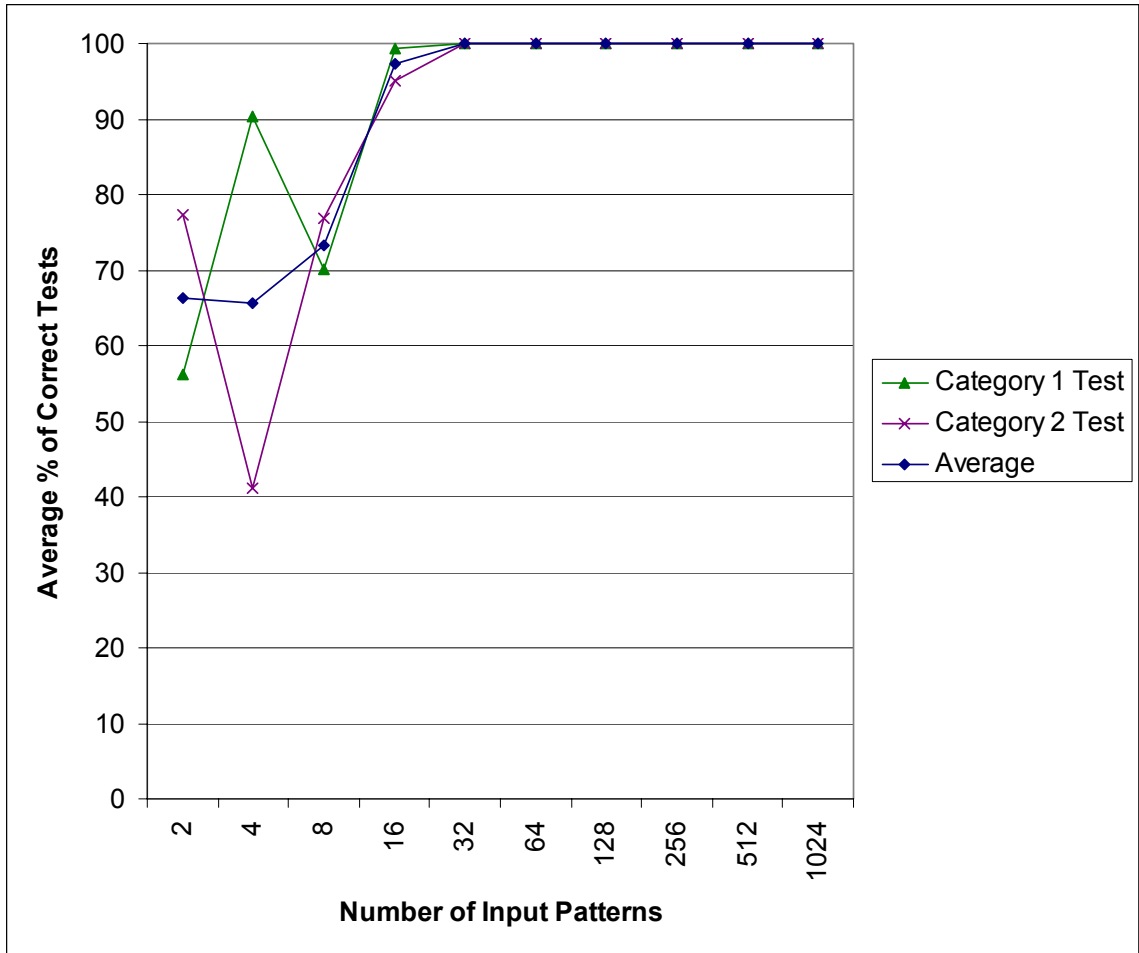


Figure 2.11 Generalization Performance by Category for One Category Unit Architecture

Number of Patterns	Number of Trials	Successful Outcomes, %	Number of Successful Outcomes	Probability of Successful Outcomes by Chance
2	25,000	51.2	12,800	0.000076
4	25,000	49.2	12,300	0.994397
8	25,000	48.0	12,000	1.000000
16	25,000	54.8	13,700	0.000000
32	25,000	48.8	12,200	0.999928
64	2,500	46.8	1,170	0.999361
128	2,500	50.8	1,270	0.217698
256	2,500	51.2	1,280	0.118997
512	2,500	55.6	1,390	0.000000
1024	2,500	62.4	1,560	0.000000

Table 2.1 Probabilities of Successful Outcomes of Zero Category Units Architecture by Chance

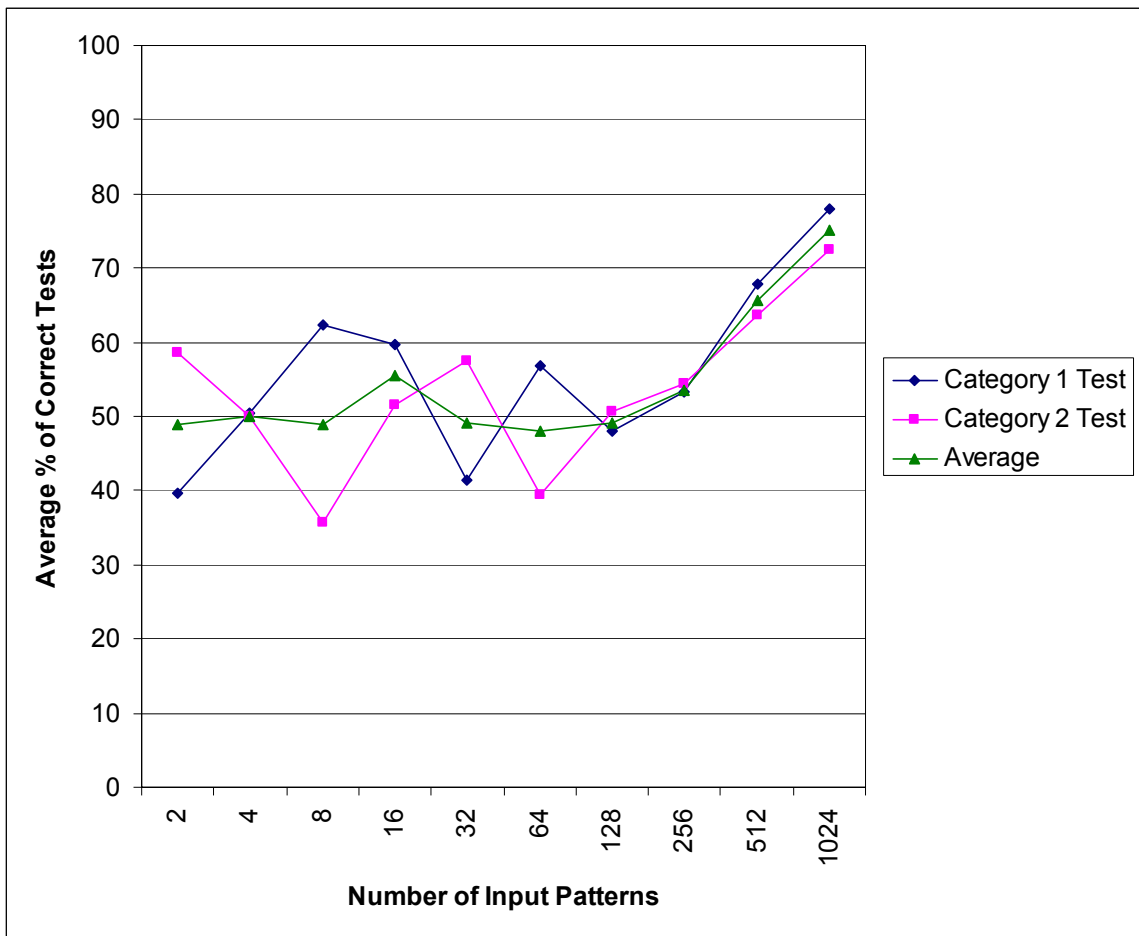


Figure 2.12 Generalization Performance by Category For Zero Category Units Architecture Simulation with 200,000 Training Epochs Limit and 0.1 Learning Rate

Chapter 3. Experiment 2A:

A More Complex Learning Task

3.1 Introduction

While categorization can be considered to be a type of association learning, associations can also be more complex than those in Experiment 1. For example, Figure 2.3 (page 22) illustrated the case of associations consisting of arbitrary patterns. In this case, associations are placed into two categories, where one of the categories has an even number of 1's in both $pattern_1$ and $pattern_2$, and the other has an odd number of 1's in both patterns. The associations between the patterns are otherwise arbitrary, i.e., $pattern_2$ cannot otherwise be predicted from $pattern_1$. The goal of the present experiment is to extend the hypotheses of Experiment 1 to more complex associations between patterns. Our main question remains the same as in Experiment 1: Does providing additional, pre-categorized input affect the learning rate or generalization ability of association learning? The stimulus change we make relative to Experiment 1 is to increase the complexity of associations.

In Experiment 2 we use a different schema for constructing $pattern_1$'s and $pattern_2$'s. Figure 3.1 shows the schema for constructing associations for Experiment 2. Each pattern ($pattern_1$ or $pattern_2$) includes a contiguous sequence of bits (i.e., all 1's or all 0's) that we refer to as a *cue*. The cue provides information that may allow quick categorization of a pattern. When all bits in the pattern's cue are 1's, the pattern is in

category₁. When all bits in the pattern's cue are 0's, the pattern is in category₂. Pattern₁'s from category₁ are associated with pattern₂'s from category₁, and similarly for patterns from category₂. A cue is positioned among the random data bits at an arbitrary location. We will use a large enough number of bits in a cue to ensure that the chance probability of having a cue is essentially zero. Also we will use a large enough number of data bits in the patterns to have numerous choices of cue location in the patterns.

We justify the change of the data set from the one used in Experiment 1 by the following reasons. First, we wanted to provide category information in a way that potentially allows a neural network to discover the category more easily than categories based on parity. Cues may satisfy this in that they have contiguous bits directly relating to the category. Second, we wanted to make the function mapping category units to the output more complex than in Experiment 1 where mapping of category units to the output was relatively simple (i.e., when category bits were 1's, the desired output was 1 0, and when category bits were 0's, the desired output was 0 1). In Experiment 2, cue locations in pattern₂'s are random and cannot be predicted from the category units. Presumably this will make mapping of category units to the pattern₂'s more complex. With such a data set design, neural networks may be able to extract category information relatively easily, and yet the function mapping pattern₁ and category information to pattern₂ will be relatively complex. The data set design is aimed to answer the following two questions: 1) If there is "easy" category information in a pattern, does providing category units affect association learning? and 2) Does making mapping from category units to the pattern₂

more complex remove the difference in performance of Model₁ and Model₂ observed in Experiment 1?

In Experiment 1 we found that Model₁ does not always have similar generalization performance for the two categories used. It requires large amounts of training and large data sets to achieve relatively high generalization performance and similar generalization performance for the two categories. We also found in Experiment 1 that providing category units improved the learning rate and generalization performance. Experiment 1 was designed to present associations from both categories in random order, in a simultaneous learning task (i.e., networks were trained on patterns from multiple categories at the same time). In part A of Experiment 2 we will also assess how category units affect learning rate and generalization for simultaneous presentation of training patterns, but now using the more complex associations of Figure 3.1 (page 59). Hence, Experiment 2A uses a more complex data set, but still uses simultaneous presentation. In part B of Experiment 2 we will assess how category units affect interference, learning rate and generalization for *sequential* presentation of training patterns of the Figure 3.1 associations.

The patterns for this experiment were designed to establish a more complex relationship for learning. This complexity occurs in the data bits of both patterns which are random but fixed, as well as locations of cues in both patterns which are also random, but fixed. These complex patterns will let us evaluate the performances of Model₁ and Model₂ and compare them with the goal of showing that Model₂ is at least not worse than Model₁ when data sets are complex and the data presentation is simultaneous.

Experiment 2A will test the following hypotheses with Model₁ and Model₂ (Figure 2.1, page 20, Figure 2.4, page 24) instantiated as neural networks:

1). When Model₁ learns associations, the generalization performance will be approximately the same for pattern₁, pattern₂ pairs of the two categories for larger data sets, but not for smaller data sets. This prediction is based on the Experiment 1 results.

2). When Model₂ learns associations, the generalization performance will be approximately the same for pattern₁, pattern₂ pairs of the two categories. This prediction is also based on the Experiment 1 results.

3). Model₂ will have approximately the same performance as Model₁ in regard to learning rate and generalization. We expect no systematic difference in learning rate and generalization in Model₁ and Model₂ because the data bits of pattern₂ cannot be predicted from pattern₁ in this experiment.

3.2 Methods

In this experiment we implemented Model₁ and Model₂ as neural networks. The neural network architecture was similar to that used in Experiment 1 (see Figure 2.6, page 36).

3.2.1 Neural Network Architecture

We used backpropagation feedforward neural networks with one layer of hidden units. The Model₁ networks had 100 input units, 50 hidden units and 100 output units. The input units for the Model₁ network architecture correspond to the 100 bits of pattern₁. Model₂ had 101 to 104 input units, 50 hidden units, and 100 output units. The input units

for Model₂ correspond to the combined pattern₁ and category vectors. The output units for both Model₁ and Model₂ correspond to pattern₂.

A constant learning rate of 0.45 was used for both Model₁ and Model₂ network architectures. The criterion for termination was that at least 85% of training patterns had a per sample error (ε) of 10% or less or that 10,000 training epochs had passed. Per sample error, ε , $0 \leq \varepsilon \leq 1$, was a weighted linear combination of three error terms (Equation 3.4). The neural network output units produced real valued activations. An output unit activation was considered correct when the absolute value of its difference with the target activation was not greater than 0.1. The first error term, $\varepsilon_{density}$ (Equation 3.1), $0 \leq \varepsilon_{density} \leq 1$, determined output error on the basis of the number of expected cue bits (i.e., 1's for category₁ or 0's for category₂) in a sequence of 20 contiguous bits. The sequence of 20 contiguous bits with the highest density of the expected bit was taken as the cue generated in the network's output. The second error term, $\varepsilon_{misplacement}$ (Equation 3.2), $0 \leq \varepsilon_{misplacement} \leq 1$, considered the error of misplacing a cue generated in the output. We calculated $\varepsilon_{misplacement}$ as the difference between positions of the left ends of the cue generated in the output and the position of the expected cue over the number of possible cue positions. The position of the left end of the cue generated in the output was determined when computing $\varepsilon_{density}$. If there were multiple sequences of 20 bits which had maximum density, we randomly chose one of the sequences as the cue generated in the output pattern. The third error term, ε_{data} (Equation 3.3), $0 \leq \varepsilon_{data} \leq 1$, considered the error in data bits and was computed as the Euclidean distance between actual output data

bits and target output data bits (without consideration of cue bits). For each of these error terms 0 indicates no error, and 1 indicates the most error possible.

$$\varepsilon_{density} = 1 - \max(S) \quad (3.1)$$

$$\text{where } S = \left\{ d \mid d = \frac{\text{number of actual cue bits}}{\text{cue length}} \forall \text{possible cue positions} \right\}$$

$$\varepsilon_{misplacement} = \frac{|\text{actual cue position} - \text{expected cue position}|}{\text{number of possible cue positions}} \quad (3.2)$$

$$\varepsilon_{data} = \text{EuclideanDistance}(\text{data bits vector}, \text{target data bits vector}) \quad (3.3)$$

$$\varepsilon = 0.5\varepsilon_{data} + 0.25\varepsilon_{density} + 0.25\varepsilon_{misplacement} \quad (3.4)$$

3.2.2 Data Sets

Data sets for this experiment contained pairs of pattern₁'s and pattern₂'s. The pattern₁'s and pattern₂'s were constructed using the schema in Figure 3.1. Each pattern (pattern₁ or pattern₂) had 80 random data bits and a cue of 20 contiguous bits at an arbitrary location among the data bits. When all cue bits were 1's, the pattern was from category₁. When all cue bits were 0's, the pattern was from category₂. The chance probability of 20 out of 20 bits being all 1's or all 0's is 0.000001 (cumulative binomial, $p_{success} = 0.5$). The data bits of a pattern had at most 12 contiguous bits of the same value. Category input units, where applicable, were set to 1 for category₁ and to 0 for category₂. Pattern₁'s were associated with an arbitrary pattern₂ from the same category except for the following constraints: a) pattern₂ was different from pattern₁ for a particular association, b) all pattern₁'s were unique, and all pattern₂'s were unique.

We assessed the effects of two variables on the performance of association learning: the number of associations, and the number of category units in the network. We used 2, 4, 8, 16, 32, 64, 128 and 256 unique $pattern_1$, $pattern_2$ pairs for training the networks, and 250 unique $pattern_1$, $pattern_2$ pairs to test generalization. Half of the training patterns were from $category_1$, and the other half from $category_2$. Also, half of the test input patterns were from $category_1$, and the other half from $category_2$. To ensure a balanced data set design, data sets with 32 or more input patterns were generated using blocks of 32 contiguous patterns, where a block contained 16 $category_1$ samples and 16 $category_2$ samples, and within a block samples were randomly ordered. A total of ten simulation runs was used for every combination of input pattern and category units, and in each simulation run a network was initially assigned randomly different weights. We did not use more than ten simulation runs for each combination due to the substantial computational time required for the simulations (e.g., ten simulation runs for 128 training patterns and a fixed number of category input units took approximately eight hours). We also did not run simulations with 512 and more training patterns for the same reason (e.g., ten simulation runs for 512 training patterns and a fixed number of category input units took approximately 88 hours). The computer hardware used for simulations was based on 2.8 GHz Intel Pentium 4 processors and 2 GB of RAM memory.

3.2.3 Performance Measures

Two performance measures were used to assess association learning: duration of training and generalization. Duration of training was measured as the number of training epochs until the performance criterion or 10,000 epochs was reached. Mean number of

training epochs across the ten simulations for each combination of training patterns and category units was used as the measure of training duration. We assessed generalization by the ability of a network to associate new pattern₁'s with the correct output pattern₂. We used the mean per sample error (ϵ) across the ten simulation runs for each combination of the number of training patterns and category units for the test associations as the measure of generalization.

3.3 Results

Figures 3.2 (page 60) and 3.3 (page 61) summarize the results of training and give the number of training epochs. Data are presented as a function of the number of input patterns, and the number of category units in the network architecture. Figure 3.2 presents number of training epochs for data sets with 64 and fewer input patterns, and Figure 3.3 shows the results of training for all data points. The simulation results showed that all architectures required approximately the same number of training epochs and there was some variation for data sets with 64 training patterns. All trials with all architectures for 128 and 256 training patterns data sets reached the maximum training epochs limit (i.e., 10,000 epochs) and have a large number of patterns with per sample error greater than the criterion (at least 56.25% for 128 training patterns data sets, and 100% for 256 training patterns data sets).

Figure 3.4 (page 62) shows the average per sample error (ϵ) for training patterns. The data are presented as a function of number of training patterns and category units. All architectures reached the 0.1 per sample error criterion for data sets with 64 and fewer training patterns. All architectures also had less than 0.11 average per sample error for

data sets with 128 training patterns. The per sample error for 256 training patterns data sets ranged from 0.1796 for the four category units architecture to 0.1909 for the zero category units architecture.

Figure 3.5 (page 63) presents the individual error terms ($\epsilon_{density}$, $\epsilon_{misplacement}$, and ϵ_{data}) and per sample error, ϵ , as a function of the number of training patterns for the zero category units architecture. The zero category units architecture errors are presented because they are representative of the results for all architectures. $\epsilon_{misplacement}$ had approximately the same value for all data sets (except for the data sets with 256 training patterns where it was higher due to incomplete training). ϵ_{data} had a non-symmetric U-shape with the smallest error for training sets at 16 patterns, and higher ϵ_{data} with greater numbers of input patterns. It is not clear why ϵ_{data} was decreased from 2 to 8 patterns, however it seems that ϵ_{data} increased from 16 to 256 training patterns due to the increasing learning task difficulty. Learning task difficulty increased roughly proportional to the increase in the number of training patterns because each $pattern_1, pattern_2$ pair had unique data bits and therefore all pairs in a training set were different. $\epsilon_{density}$ had a roughly non-symmetric inverse U-shape with the highest error for training sets with 16 patterns, and smaller $\epsilon_{density}$ with greater number of input patterns. Again, it is not clear why $\epsilon_{density}$ was low for training sets with less than 16 patterns. It does seem clear though that $\epsilon_{density}$ decreased with training sets with more than 16 input patterns because with more input patterns of the same type (e.g., $category_1$) it was presumably easier for the network to

detect the general property that there were cues in pattern₁'s and reproduce them in the network's output.

Figure 3.6 (page 64) presents generalization test results for all architectures. With the exception of the data sets with two training patterns, architectures with category units had less per sample error than the zero category units architecture for all data sets. Also notable here is the reduction in per sample error with larger training data sets. As in training, we considered the output correct when the per sample error was less than 0.1 (see also section 3.2.1 Neural Network Architecture). Per sample error in generalization tests was in general exceeding the 0.1 criterion for all generalization test patterns for all architectures. In other words networks generalized at relatively low performance levels and showed much worse (i.e., zero correct tests) results than in Experiment 1 (see Figure 2.9). Poor generalization was due to the complexity of data sets that had random data bits and random cue locations. Random data bits and random cue locations did not allow good generalization of training data.

Figures 3.7, 3.8, and 3.9 (pages 65-67) give the results of generalization tests with respect to $\epsilon_{density}$, $\epsilon_{misplacement}$, and ϵ_{data} error terms for all architectures. Also noticeable in Figure 3.7 is a general trend to lower error, this time in cue density, with increases in the number of training samples. This is presumably because with additional training patterns networks were able to recognize and utilize the structure of the patterns better. Another marked feature of Figure 3.7 is that except for the data sets with two training patterns, architectures with category units generally had less cue density error than the zero category units architecture. A paired sample t-test showed that the zero category units

architecture had statistically significantly larger $\varepsilon_{density}$ values than any of the architectures with category units (one-tailed, $t(5) = 6.05, 2.34, 4.91, 5.55$; $p < 0.05$; excluding data points for two and four training patterns). It appears that the networks with category units learned the rule: generate a contiguous sequence of the same bits as the category units in the output. We did not consider in the t-tests data points for training sets with two and four training patterns because they did not provide reliable results due to the small number of training patterns (only two of four one-tailed tests with all data points showed statistically significant differences). The smallest $\varepsilon_{density}$, 0.13064 (four category units architecture, 256 training patterns), corresponds to 17 correct cue bits in the output. The largest $\varepsilon_{density}$, 0.68834 (two category units, eight training patterns), corresponds to 6 correct cue bits in the output.

The $\varepsilon_{misplacement}$ error term for all architectures ranged from 0.2944 (24 bits placement error) for the four category units architecture and 32 training patterns to 0.393095 (32 bits placement error) for the four category units architecture and four training patterns¹. By chance, we would expect $\varepsilon_{misplacement}$ to have $\mu = 40$ bits and $\sigma = 23.09$ bits (given a uniform distribution $\sim U(0,80)$). There were no noticeable relationships between $\varepsilon_{misplacement}$ and the number of training patterns or number of category units in the network's architecture. $\varepsilon_{misplacement}$ contributed 7.36 to 9.83% of the

¹ Only means were recorded for $\varepsilon_{misplacement}$ and so a t-test to compare $\varepsilon_{misplacement}$ values to the mean expected by chance was not possible.

per sample error, ε , in generalization. Such a level of error in the placement of cues in the output is due to an arbitrary cue location in the pattern₂, independent of pattern₁ or category bits. There is, in fact, no way for correct generalizations to occur with placement of the cue.

The ε_{data} error term, presented in Figure 3.9, was in most cases lower for the architectures with category units compared to the zero category units architecture. Only for data sets with small numbers of training patterns (two and four) and when training was incomplete (256 training patterns) was ε_{data} for architectures with category units higher than the error in data for the zero category units architectures. A paired sample t-test statistically showed that the zero category units architecture had significantly larger ε_{data} than the architectures with category units (one-tailed, $t(5) = 2.06, 3.12, 5.32, 2.677$, $p < 0.05$). We did not consider in the t-tests data points for training sets with two and four training patterns because they did not provide reliable results presumably due to a small number of training patterns (only two of four one-tailed t-tests with all data points showed statistically significant differences).

Figure 3.10 (page 68) presents generalization test results with respect to each category for the zero category units architecture. Only results from the zero category units architecture generalization tests are presented because all architectures show similar results. Errors for both categories have some degree of variance and differ at most by 3.4%. Though the differences are small, they are likely to be due to the criterion of 85% of training patterns with per sample error less than 0.1. This criterion of terminating

training may have resulted in patterns from the individual categories having slightly different errors that then created small differences in the generalization test results.

3.4 Discussion

We assessed differences for Model₁ vs. Model₂ in a simultaneous learning task when pattern₁'s and pattern₂'s form complex associations. Learning rate was measured as the number of training epochs. Generalization was assessed on a set of 250 novel patterns with equal number of patterns from both categories using per sample error. Per sample error was a weighted sum of three error terms: cue density error, cue misplacement error and data error. Cue density error assessed differences of generated cue bits from expected bits. Cue misplacement assessed error in generated cue locations from expected locations based on the distance in bits. Data error used Euclidean distance to compute errors in generated data bits. Results of the experiment showed that category units in the input reduced cue density error and data error somewhat. Misplacement error had approximately the same level for both models.

Considering our hypotheses, based on the results of this experiment, we can state that:

- 1). Different from our expectations, Model₁ had approximately the same generalization performance for pattern₁, pattern₂ pairs from both categories for all training sets (see also Figure 3.10, page 68). This result shows that generalization performance of a network with respect to patterns from a particular category is dependent not only on the number of training patterns, but also on the complexity of data and mapping functions.

2). As expected, Model₂ had approximately the same generalization performance for pattern₁, pattern₂ pairs from the two categories used.

3). As expected, Model₂ had approximately the same learning rate as Model₁ (see Figures 3.2, page 60, and 3.3, page 61), but different from our expectations, Model₂ had better generalization than Model₁ (see also Figure 3.6, page 64). These improvements in generalization performance in Model₂ vs. Model₁ were due to the use of category units. Better generalization results for Model₂ show that providing additional category information may improve association learning, even when category information in the data is relatively explicit. Without considering results of generalization tests for small data sets (two and four training patterns), Model₂ always had less error than Model₁.

Improvement of the per sample error in Model₂ was due to improvements in two of its components, cue density error and data error. The other component, cue misplacement error, was approximately at the same level for both models. Cue misplacement error had some variance, but did not generally change as a function of the number of training patterns (see also Figure 3.8, page 66) which was not surprising due to the random placement of cue bits in pattern₁ (relative to pattern₂) this placement could not be generalized. Both Model₁ and Model₂ architectures showed a trend of lower per sample error with increases in the number of training patterns.

Results of this experiment showed that additional precategory input tended to improve generalization performance of association learning when category information in the data is relatively explicit. Arguably high generalization error levels (i.e., 45-56%) are presumably due to the complexity of the data sets. As we discussed in the Introduction to

this chapter, associations constructed by the schema in Figure 3.1 are aimed to provide category information in a relatively explicit manner (i.e., in the cue bits), but require a complex mapping from input to output. Cue density error in generalization tests for the zero category units architecture drops by 0.49 when the number of training patterns increased from eight to 256 (see Figure 3.7, page 65). This is a relatively large reduction of error (almost a half of its range) and shows that even without category units, cues were easily found by the neural networks. Cue misplacement error was relatively high and ranged approximately from 0.3 to 0.4 in generalization tests due to random cue locations in patterns. Data error was the largest error term that increased as a function of the number of training patterns. These results with data error are also due to random data bits in patterns. Such results with respect to the error terms showed that data sets provided relatively easily accessible category information with complex mapping from input to output.

Overall, results showed that associations constructed by using the schema in Figure 3.1 (page 59) provide relatively explicit category information and require learning a complex function mapping input to output. Results of the experiment also showed that additional explicit category information may provide improvement in association learning even when category information is contained in patterns in an easy to extract form and the relation between $pattern_1$ and $pattern_2$ is complex.

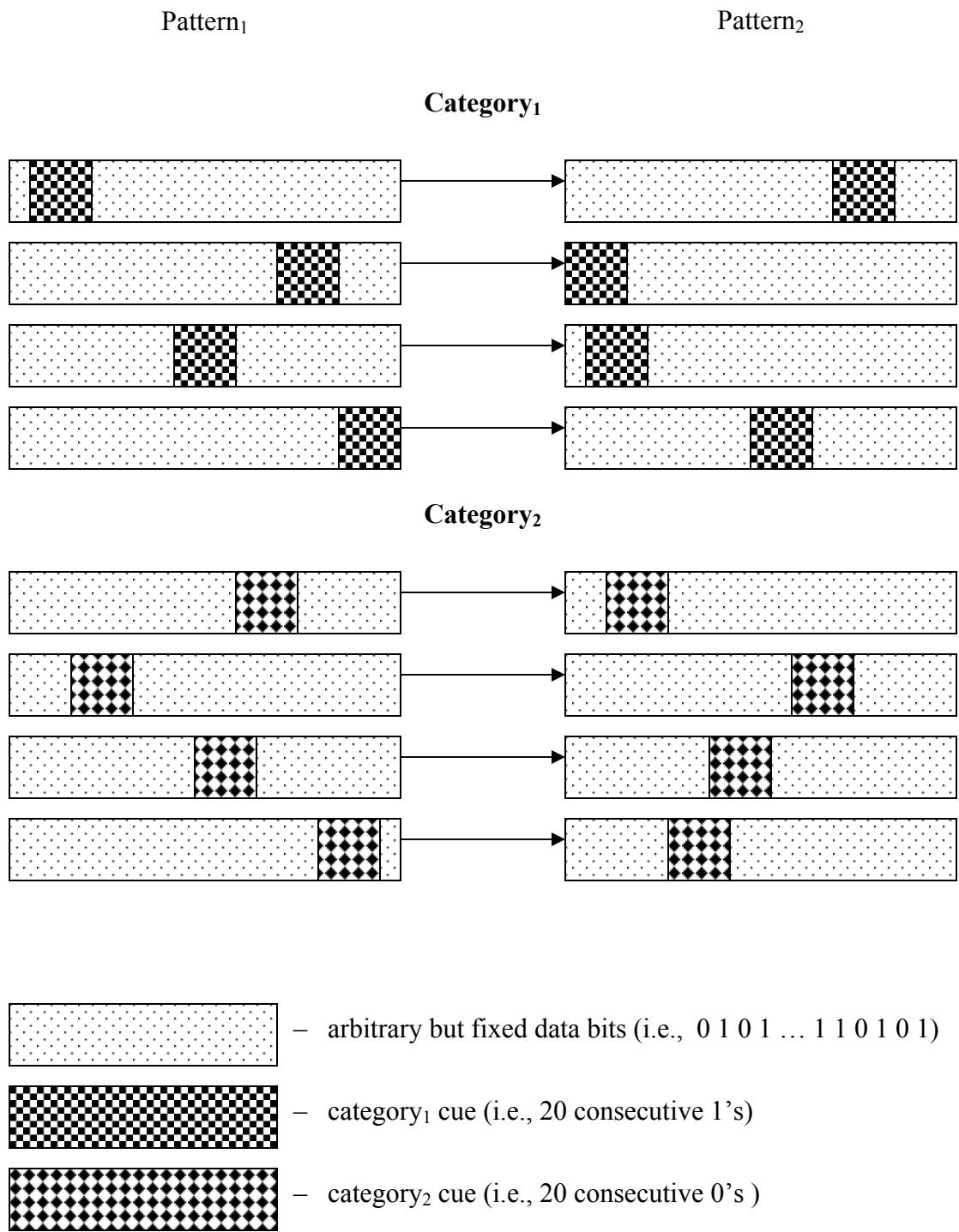


Figure 3.1 Schema of Associations to be Learned in Experiment 2

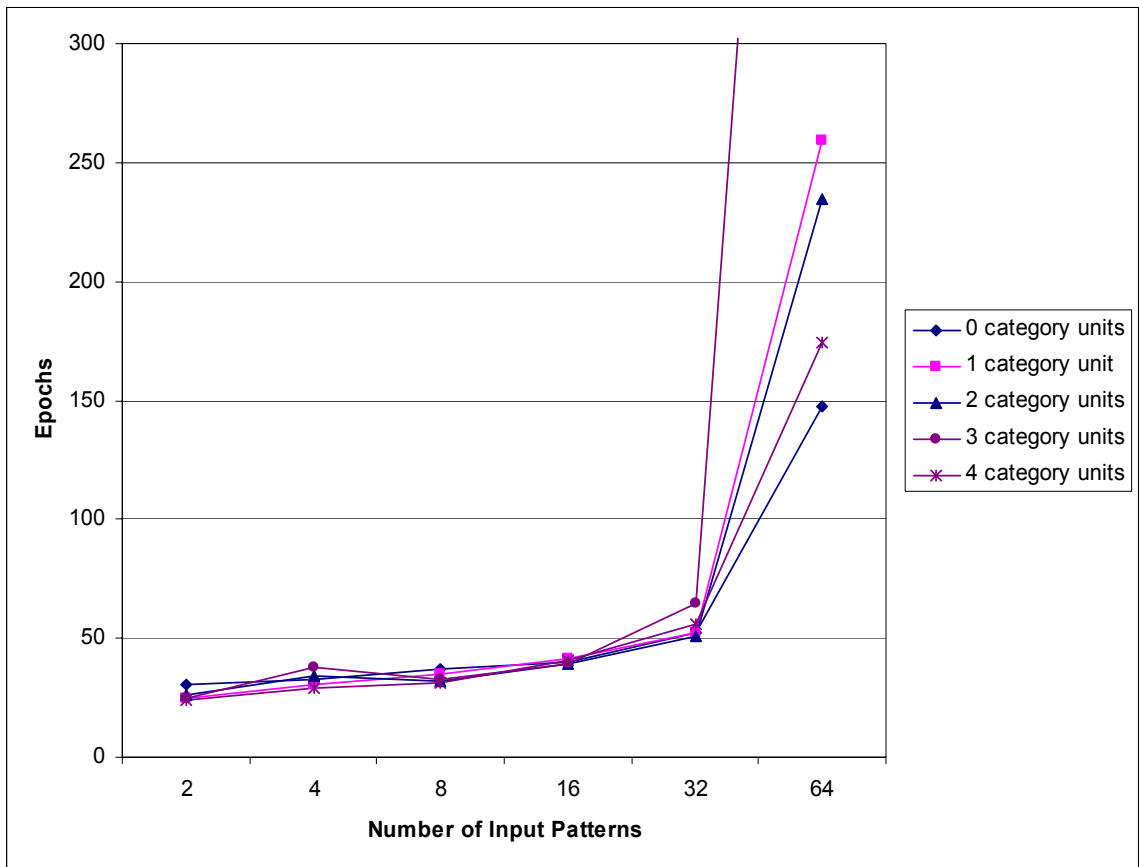


Figure 3.2 Amount of Training Required for All Architectures

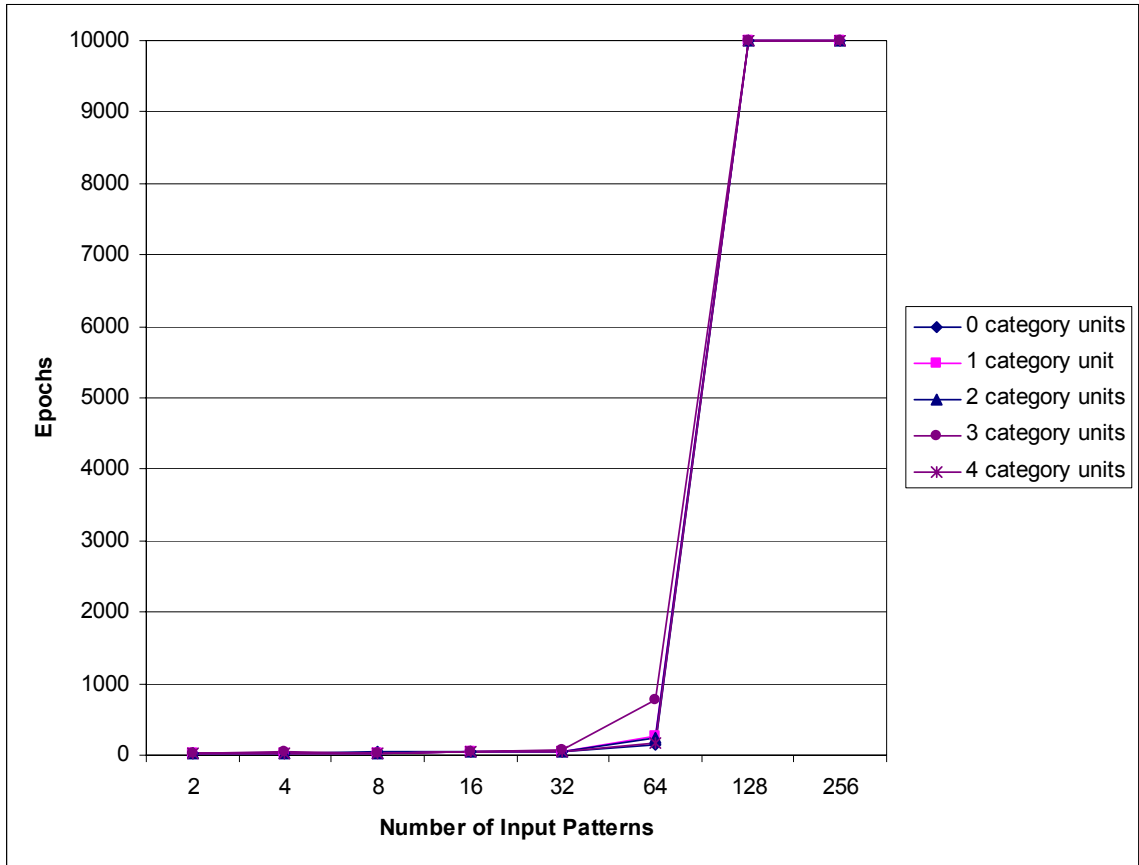


Figure 3.3 Amount of Training Required for All Architectures (for all data points)

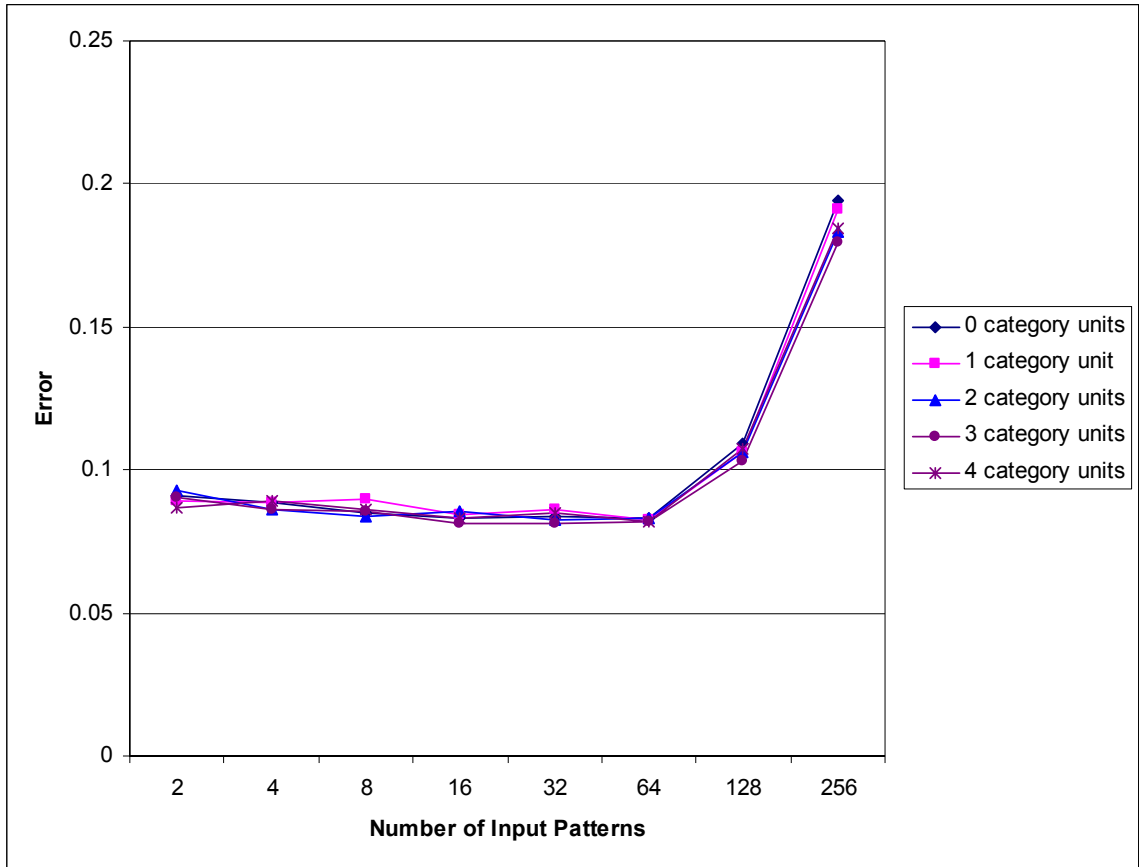


Figure 3.4 Per Sample Error for All Architectures (Training Results)

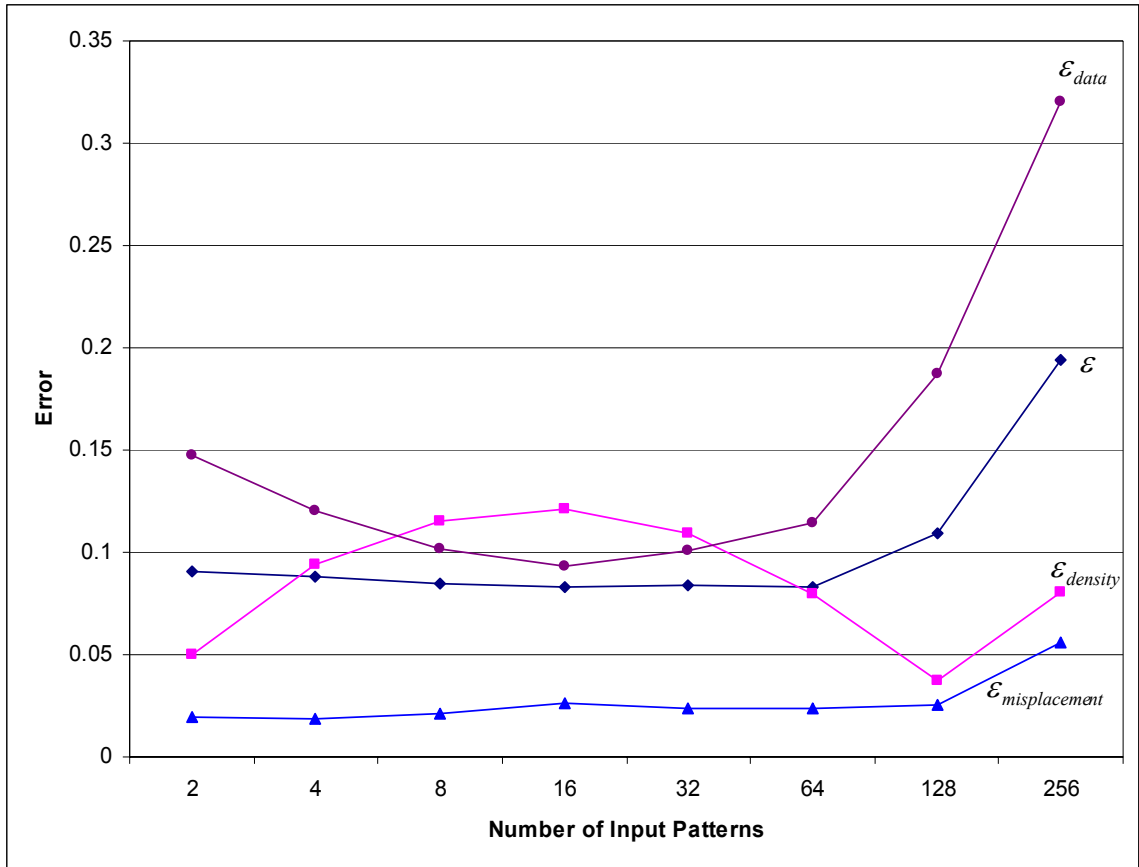


Figure 3.5 Per Sample Error Terms for Zero Category Units Architecture (Training Results)

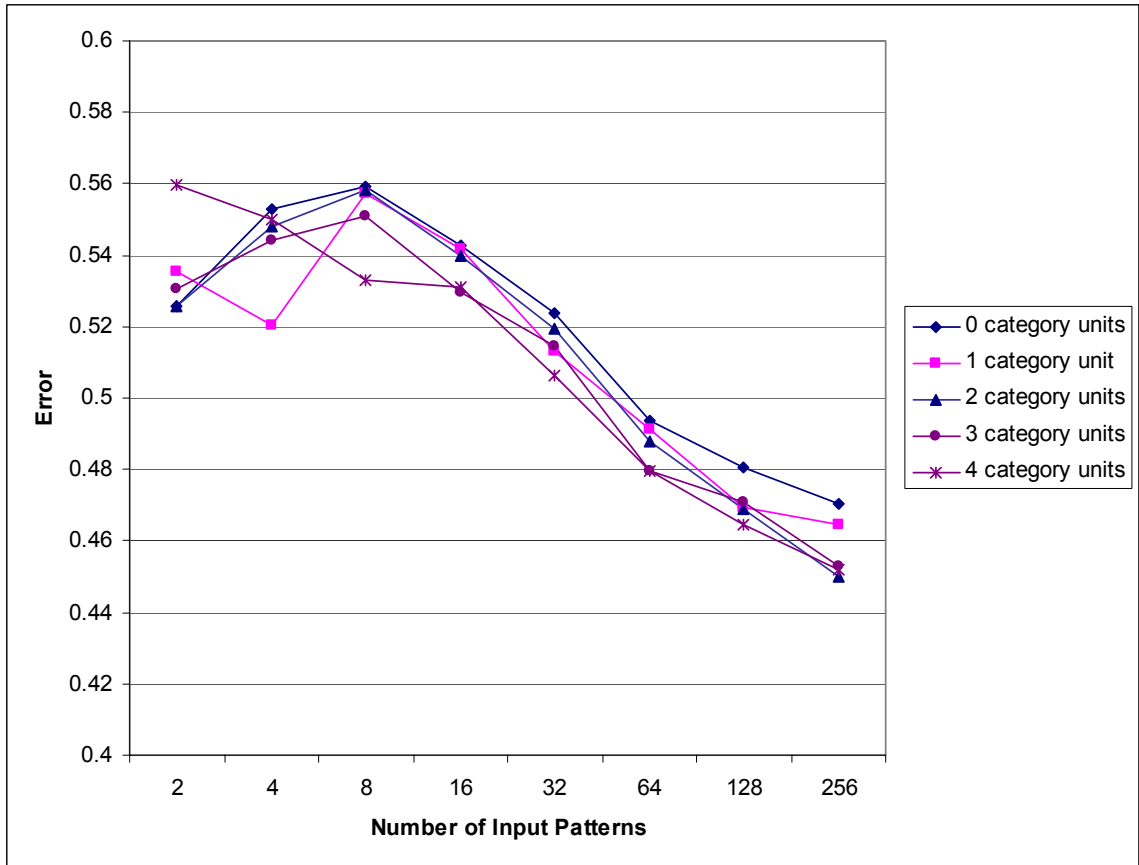


Figure 3.6 Per Sample Error, ε , for All Architectures (Generalization Tests)

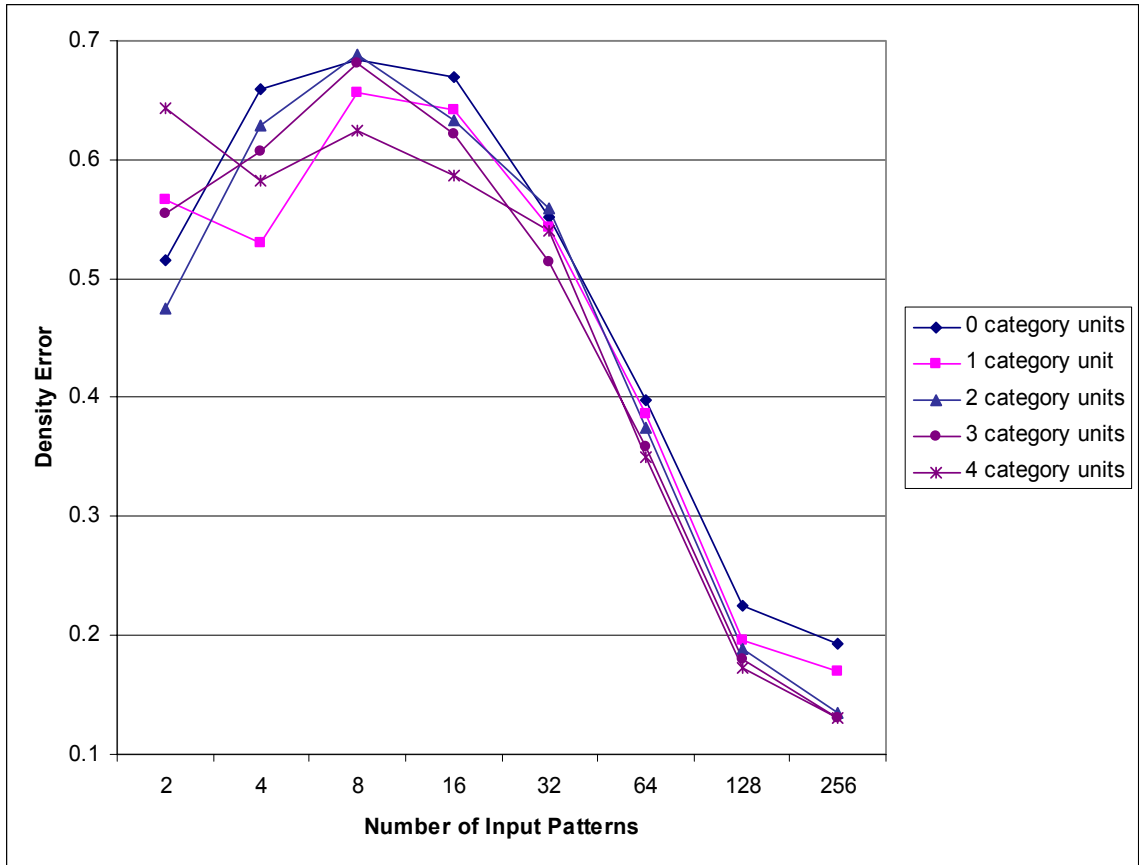


Figure 3.7 $\varepsilon_{density}$ for All Architectures (Generalization Tests)

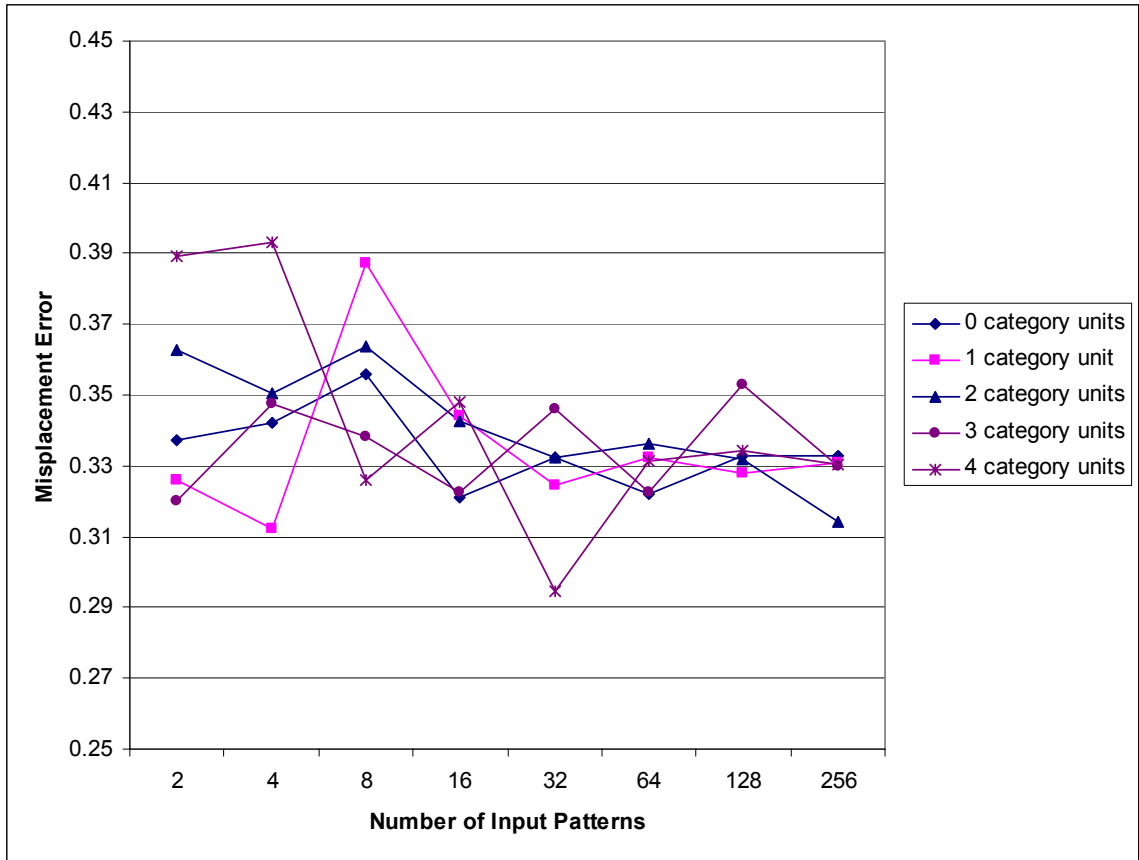


Figure 3.8 $\epsilon_{misplacement}$ for All Architectures (Generalization Tests)

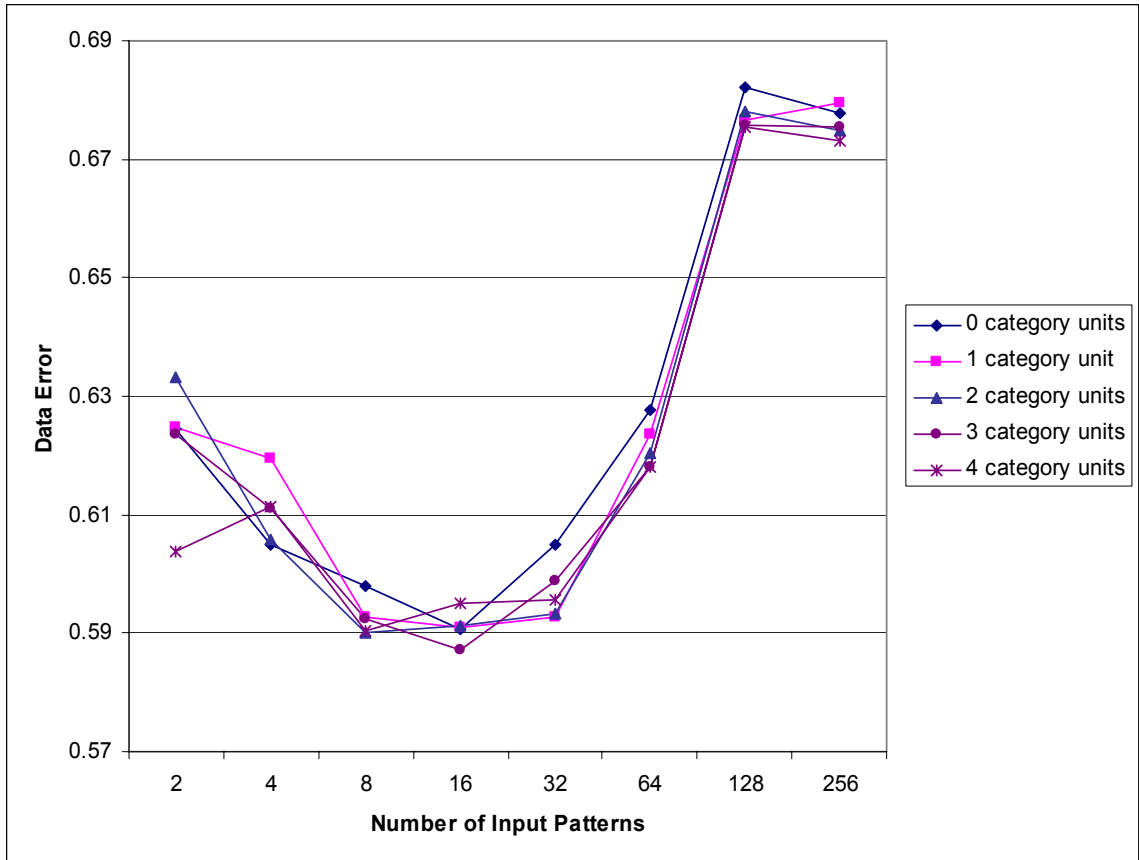


Figure 3.9 ϵ_{data} for All Architectures (Generalization Tests)

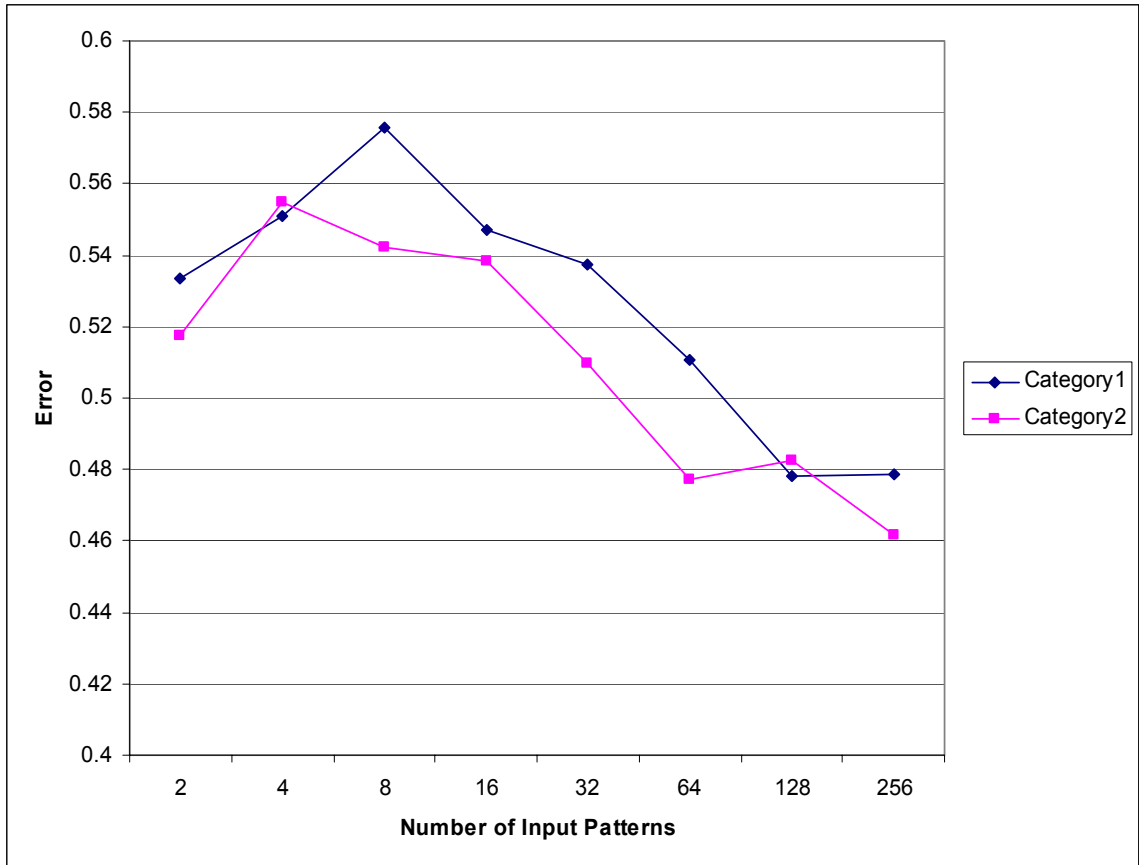


Figure 3.10 ε by Category for Zero Category Units Architecture (Generalization Tests)

Chapter 4. Experiment 2B:

A Sequential Learning Task

4.1 Introduction

Experiments 1 and 2A showed that category information may, to some degree, increase the performance of association learning models. In both experiments training patterns were presented in a simultaneous manner. That is, training patterns from both categories were interspersed in the training sequence. Experiment 2B is aimed to explore the effects of category information on interference in association learning, in a sequential learning task. In a sequential learning task a network is first trained on examples from one category (e.g., category A), and then it receives training on examples from another category (e.g., category B). A typical outcome of sequential tasks for connectionist models is loss of performance on the initial training. In this case, we might expect reduced subsequent performance on category A after the training on category B examples. Although in Experiment 2B the learning task is different from that in Experiments 1 and 2A, we will use similar performance measures to assess interference. In this experiment we will first train networks on category₁ patterns, and then train them on category₂ patterns. We will assess interference by comparing training and generalization performance on associations from category₁ prior to and after learning associations from category₂. In a similar way, we will separately reverse the order of training and assess interference when the initial training is on category₂ associations. Our

main question remains the same as in Experiments 1 and 2A: Does providing additional, pre-categorized, input affect the learning rate or generalization ability of association learning? In the context of a sequential learning task, we are also interested in the effects of additional, pre-categorized, input on interference in association learning.

As in Experiment 2A, in Experiment 2B we will also use associations constructed according to the schema in Figure 3.1 (page 59). Each pattern (pattern_1 or pattern_2) includes a cue at an arbitrary location among data bits of the pattern. The cue is a contiguous sequence of bits with the same value marking the category of the pattern. If the cue bits are 1's, then the pattern is from category₁. If the cue bits are 0's then the pattern is from category₂. Pattern_1 's are associated with arbitrary pattern_2 's from the same category, as in Experiment 2A. Data bits of patterns are random bits with 12 or less contiguous bits of the same value. Presumably, associations constructed in this way provide relatively explicit category information in the form of cues, and at the same time provide a rather complex mapping function of pattern_1 and category information to the pattern_2 due to the arbitrary locations of cues in patterns and random data bits (see the Chapter 3 Introduction for more details).

In Experiment 2A we found that Model₂ had somewhat improved performance compared with Model₁. Model₂ had lower cue density error and lower data error in generalization. The improvements were due to category information provided with pattern_1 's as an input. We also found in Experiment 2A that performance across the two categories of associations was approximately the same in both models. In this experiment

we assess how additional category information affects learning rate, interference and generalization performance of association learning in a sequential learning task.

Experiment 2B tested the following hypotheses with Model₁ and Model₂ (Figure 2.1, page 20, Figure 2.4, page 24) instantiated as neural networks and trained in a sequential manner:

- 1). When Model₁ learns associations, the performance will be approximately the same for pattern₁, pattern₂ pairs of the two categories.
- 2). When Model₂ learns associations, the performance will be approximately the same for pattern₁, pattern₂ pairs of two categories.
- 3). Model₂ will have approximately the same performance as Model₁ with regard to learning rate.
- 4). Model₂ will have some improvement in generalization.
- 5). Model₂ will have less interference compared to Model₁.

In the third hypothesis we mean that the training time required for training on patterns from the first training set and from the second training set should be approximately the same pairwise for both models. We make this prediction based on Experiment 2A results which showed that category information (i.e., Model₁ vs. Model₂) did not affect training time.

We base the fourth hypothesis on the results of Experiment 2A which showed some improvement in generalization in Model₂ compared to Model₁. The improvement in generalization was due to lower cue density error and data error. Such improvement in

generalization may also occur in Model₂ in the present experiment when the only change from Experiment 2A is the manner of presenting training patterns.

Regarding the fifth hypothesis, we predict, based on the lower cue density error and lower data error observed in Experiment 2A that Model₂ may have reduced interference in a sequential learning task in neural network models as compared to Model₁.

4.2 Methods

As in the previous experiments, we implemented Model₁ and Model₂ as neural networks. The neural network architectures were as described in Experiment 2A (see Section 3.2).

4.2.1 Neural Network Architecture

Neural network architectures and parameters were the same as in Experiment 2A (see Section 3.2.1). However, the simulation procedure was different than that in Experiment 2A to provide assessment for interference. Each simulation run had two parts. The first part of a simulation run, which we call Case α , consisted of the following steps:

- A. Initialize a network with randomly different weights.
- B. Train the network on the category₁ training set, and conduct category₁ training evaluation.
- C. Test generalization on the category₁ test set (“initial generalization”).

- D. Train the network on the category₂ training set, and conduct category₂ training evaluation.
- E. Test performance on the category₁ training set (“recall”).
- F. Test generalization on the category₁ test set (“recall generalization”).
- G. Test generalization on the category₂ test set (“other set generalization”).

The second part of a simulation run, which we call Case β) consisted of the above steps with “category₁” and “category₂” reversed (i.e., in Case α a network is trained first on category₁ and in Case β a network is trained first on category₂). We call Step E evaluation *recall* because we are assessing performance on one category (e.g., category₁) after intervening training on another category (e.g., category₂).

4.2.2 Data Sets

Data sets for this experiment contained pairs of pattern₁'s and pattern₂'s. Patterns were constructed using the schema in Figure 3.1 (page 59) in the same way as in Experiment 2A (see Section 3.2.2).

We assessed the effects of three variables on the performance of association learning: the number of training associations, the number of category units in the network, and the training order of categories (i.e., Case α or Case β). We used 2, 4, 8, 16, 32, 64, 128 and 256 unique pattern₁, pattern₂ pairs for training a network, and 250 unique pattern₁, pattern₂ pairs to test generalization. Training patterns were organized into two sets of patterns. The category₁ training set contained pattern₁, pattern₂ training pairs only from category₁. The category₂ training set contained pattern₁, pattern₂ training pairs only from category₂. The category₁ and category₂ training sets contained an equal number of

pattern pairs. In a similar way, the category₁ test set consisted only of test pairs of patterns from category₁, and the category₂ test set consisted only of test pairs of patterns from category₂. Testing sets also contained equal numbers of pattern pairs from the two categories. A total of ten simulation runs was used for every combination of the above three variables. We did not use more than ten simulation runs and we did not run simulations with more than 256 training patterns due to the substantial computation time requirements (e.g., ten simulation runs for 256 training patterns and a fixed number of category units required eighteen hours). The computer hardware used for simulations was based on 2.8 GHz Pentium processors and 2 GB of RAM memory.

4.2.3 Performance Measures

We assessed generalization by the ability of a network to associate new pattern₁'s with the correct output. As a measure of generalization for category₁ and category₂ test sets we used the mean per sample error across the ten simulation runs for each combination of training patterns, category units and training order of categories for the test sets of associations. Per sample error, and its three components (cue density error, misplacement error and data error) were defined in Section 3.2.1. We assessed interference by comparing recall performance (Step E) with performance on the initial training set (Step B). We used mean per sample error across ten simulation runs for each combination of training patterns, category units and training order of categories as a measure of recall performance for the initial training set. We also assessed the duration of training for each training set. It was measured as the number of epochs until the performance criterion or 10,000 epochs was reached. Training duration was measured as

mean number of training epochs for each set across ten simulation runs for each combination of training patterns, category units and training order of categories.

4.3 Results

Figures 4.1 (page 84) and 4.2 (page 85) summarize the results of initial training (Step B) in the sequential learning task and give the number of training epochs. Data are presented as a function of the number of input patterns, and the number of category units in the network architecture. Initial training (Step B) showed similar behavior in Cases α and β (first training on category₁ and category₂ respectively). All architectures had some variation in both Cases α and β . Figures 4.1 and 4.2 show that with increasing numbers of input patterns (e.g., 128 and more) training time rapidly increased. Initial training (Step B) in all simulation runs with 256 input patterns reached the 10,000 epochs limit. Results of the initial training (Step B) did not reveal clear distinctions between the training time required by architectures with and without category units and were similar to the results in Experiment 2A (see Figures 3.2, 3.3).

Figure 4.3 (page 86) presents results of the subsequent training (Step D) for all architectures in Case α . Data are presented as a function of the number of input patterns and the number of category units in the network architecture. Figure 4.3 is also representative of subsequent training (Step D) for all architectures in Case β . All architectures had some variation in the number of required training epochs, and all architectures reached the 10,000 epochs limit with the 128 and 256 training patterns data sets. The results did not show clear distinctions between architectures with and without category units.

The data for the zero category units architecture from Figure 4.1 and 4.3 are represented in Figure 4.4 (page 87) to enable direct comparison between initial training times (Step B) and subsequent training times (Step D). The results shown in Figure 4.4 are representative of all architectures, both Cases α and β . Although it appears that the subsequent training time (Step D) took a greater number of epochs than the initial training (Step B) time, paired t-tests at a 0.05 significance level did not show statistically significant differences between initial (Step B) and subsequent (Step D) training times in general for the architectures in Cases α and β comparing zero category units architectures to the other architectures. We did not consider 128 and 256 input patterns data points in the t-tests because training was not completed with those data sets.

Figure 4.5 (page 88) summarizes results of the initial (Step B) and subsequent training (Step D), Case α for the zero category units architecture in terms of per sample error (ϵ). Data from the zero category units architecture simulations are representative of the results for all architectures, and Cases α and β . Both initial (Step B) and subsequent (Step D) training results showed approximately the same level of per sample error and this arises due to the 0.1 per sample error criterion. The data for 256 training patterns showed some variation which is presumably due to the incomplete training.

Figure 4.6 (page 89) presents per sample error (ϵ) for the initial training (Step B) and recall tests (Step E) for the zero category units architecture, Case α . The data are representative of all architectures, and Cases α and β . The main result of recall tests was that architectures with and without category units had greater recall error than the error

after completion of the initial training (Step B). This shows that both Model₁ and Model₂ suffered from interference.

Figure 4.7 (page 90) presents recall tests per sample error terms for the zero category units architecture, Case α . Data in Figure 4.7 are also representative of the recall test results for the zero category units architecture, Case β . The data error term was increasing with increases in the number of training patterns. Per sample error was generally increasing with increases in the number of training patterns.

Per sample error for recall (Step E) for the one category unit architecture are presented in Figure 4.8 (page 91) as representative of the results for architectures with category units, Cases α and β . Comparing Figures 4.7 and 4.8 shows that architectures with category units (Figure 4.8) had similar recall behavior to architectures without category units (Figure 4.7). Although the subsequent training (Step D) did not completely destroy information learned in the initial training (Step B), there was a relatively large increase of the recall error after subsequent training (Step D) that may be called catastrophic interference.

In order to quantitatively evaluate the effects of category units, training patterns and training order on recall error we performed an analysis of variance (ANOVA). We used a three factor ANOVA model. Number of category units, number of training patterns, and training order were the factors in the model and mean per sample error in recall tests was the dependent variable. These data are presented in Figures 4.9 (page 92) and 4.10 (page 93). We did not use data for 256 training patterns sets due to incomplete training with those data sets. ANOVA showed that training order and its interactions with other factors

were not statistically significant sources of variation at the 0.05 significance level. The above results comply with our expectations of approximately equal performance on category₁ and category₂ patterns stated in Hypotheses 1 and 2 (see Introduction to this chapter). Number of category units, number of training patterns, and number of category units and training patterns interaction were statistically significant sources of variation at the 0.05 significance level ($F(4,24) = 3.44, p = 0.0233$; $F(6,24) = 44.62, p < 0.0001$; $F(24,24) = 2.89, p = 0.0060$). The above results show that number of category units, training patterns and their interaction affected per sample error of Model₁ and Model₂ architectures.

The above ANOVA results showed that the number of category units accounted for some of the variance in per sample error. However, these results do not provide further detail. In specific, this analysis does not inform us about Model₁ vs. Model₂ differences. To provide further detail on Model₁ and Model₂ architectures we used statistical contrasts. In a contrast, a group of levels of a factor (e.g., one, two, three and four category units) is compared with another group of the levels of the factor (e.g., zero category units) and an F-test or t-test is computed to check the difference between the two groups for statistical significance. The contrast for assessing differences between Model₁ and Model₂ per sample error (i.e., the zero category units architecture compared to all other architectures) in recall did not show statistically significant difference at the 0.05 significance level, though we found statistically significant differences between some architectures in pairwise contrasts. In the pairwise contrasts the per sample error of the four category units architecture was statistically significantly different from the zero,

one, two and three category units architectures at the 0.05 significance level ($F(1,24) = 8.37, p = 0.008$; $F(1,24) = 9.05, p = 0.0061$; $F(1,24) = 4.32, p = 0.0486$; $F(1,24) = 9.92, p = 0.0043$). Inspecting the means shows that the difference was that the four category units architecture (mean 0.4693) had statistically significantly higher per sample error than any other architecture (means not greater than 0.4464). The contrast between the four category units architecture and the group of all other architectures also showed statistically significantly higher per sample error for the four category units architecture at the 0.05 significance level ($F(1,24) = 12.38, p = 0.0018$; $t_{0.025, 24} = -3.52, p = 0.0018$). These results with the four category units architecture show that category units had an effect of increasing interference (i.e., increasing per sample error) in recall in a sequential learning task.

Figure 4.11 (page 94) presents initial generalization test (Step C) per sample error (ϵ), Case α . Figure 4.11 is also representative of all architectures, Case β . These results are similar to the results obtained in Experiment 2A (see Figure 3.6, page 64) in the trend to lower per sample error with the increase of the number of training patterns. Unlike in the simultaneous learning task in Experiment 2A, in Experiment 2B both Model₁ and Model₂ architectures showed approximately the same per sample error in the initial generalization test results. Results of the other set generalization tests (Step G), both Cases α and β , showed approximately the same results as the data in Figure 4.11. Similar levels of generalization error for category₁ and category₂ patterns complies with our expectations of approximately equal performance for patterns from the two categories.

Figure 4.12 (page 95) presents per sample error in recall generalization tests (Step F) for all architectures, Case α . The data in Figure 4.12 are also representative of recall generalization (Step F) in Case β . Recall generalization per sample error (Figure 4.12) increased after subsequent training (Step D) compared to the initial generalization (Step C) presented in Figure 4.11. The increase in error of recall generalization shows that networks had some degree of interference in generalization. The increase of per sample error in recall generalization (Step F) was due to increases of some error terms compared to the results in Step C (Figure 4.11). Figures 4.13 (page 96) and 4.14 (page 97) present data error and cue density error for recall (Step F) and initial (Step C) generalization tests for all architectures, Case α . The data are also representative of the results in Case β . Noticeable is that data error and cue density error in general were higher in recall generalization (Step F) compared to initial generalization (Step C). The increase of data error (Figure 4.13) was due to presenting patterns with unique data bit sequences in subsequent training (Step D). The increase of cue density error (Figure 4.14) was due to subsequent training (Step D) on patterns from the other category. Figure 4.15 (page 98) presents misplacement error in recall generalization (Step F) for all architectures, Case α . The data in Figure 4.15 are representative for misplacement error in recall generalization (Step F) in Case β . Misplacement error had a relatively large variation and did not show a general increase or general decrease compared to initial generalization (Step C).

4.4 Discussion

We assessed effects of the number of training patterns, the number of category units in the input, and the training order of categories (i.e., training first on category₁ vs.

training first on category₂) on the performance of association learning in a sequential learning task when pattern₁'s and pattern₂'s form complex associations. The sequential learning task consisted of initial training (Step B) of a network on associations from one category, and subsequent training (Step D) on associations from the other category. An interference effect in a sequential learning task is said to occur when after additional training recall of a network on patterns from initial training has a relatively high level of error. The main result of Experiment 2B was that category units did not help to reduce interference in Model₂ compared to Model₁ and furthermore category units actually *increased* interference in the four category units architecture comparing to other architectures, with data constructed according to the schema in Figure 3.1.

As expected in Hypothesis 1, Model₁ showed approximately the same training time and error for associations of the two categories. Also, as expected in Hypothesis 2, Model₂ showed approximately the same training time and error for associations of the two categories.

As expected in Hypothesis 3, Model₂ had approximately the same learning rate as Model₁. Initial (Step B) and subsequent (Step D) training times were approximately the same across Model₁ and Model₂. Both models showed a trend of increasing training time with increases in the number of training patterns. Increases of training time were due to an increasing learning task complexity with the increases in the number of training patterns.

Different from our expectations (see Hypothesis 4), generalization test results showed approximately the same level of error for Model₁ and Model₂. Recall

generalization (Step F) was affected by the subsequent training (Step D) and showed increase of error compared to initial generalization (Step C). Cue density error was the most affected error term and in general was higher than in initial generalization (Step C). Data error in recall generalization (Step F) also was in general higher than in initial generalization (Step C).

Different from our expectations (see Hypothesis 5), in some cases Model₂ showed statistically significantly higher levels of interference than Model₁. Analysis of variance showed that number of category units and number of training patterns accounted for statistically significant variance. Training order (i.e., Case α vs. Case β) did not significantly account for variance which complies with our expectations of approximately equal performance on association from the two categories. Statistically significantly higher recall error was observed with the four category units architecture and was due to the number of category units. The above results show that category units may increase interference in sequential learning tasks.

In addition to our specific hypotheses, we also observed that Model₁ and Model₂ both showed relatively high levels of interference. By relatively high, we mean that the recall error was much greater than the initial training error (e.g., see Figure 4.6, page 89). This error arose from increases in cue density error and data error terms. A relatively large increase in cue density error showed that networks essentially were not able to effectively utilize the category information of input patterns (i.e., cues and category units) to distinguish pattern categories. It may be the case that while we think about cues as category markers (i.e., specialized data), the neural networks did not distinguish cues

from other data in patterns. The increase of data error was due to a relatively large number of unique data bit sequences. Data error showed an increasing trend with an increase of training patterns. Cue misplacement error was affected by the subsequent training (Step D) less than the other two error terms. This result was likely due to a relatively small number of possible cue locations in patterns.

Overall, the results of Experiment 2B showed that Model₁ and Model₂ had approximately the same training time and error levels, with a relatively large recall error in sequential learning tasks with data constructed according to the schema in Figure 3.1. We also found that category units may increase recall error. Increased recall error was observed with the four category units architecture.

Under the conditions of this experiment, category information was included in patterns in the form of cues, as well as category units where applicable. Because of the structure of the data in the patterns (see Figure 3.1, page 59), category units did not have a positive effect on training time and generalization in Experiment 2B. Experiments on data that does not include cues and provides relatively complex mappings of input pattern and category units to the output pattern may help to further assess effects of category information on association learning and interference in sequential learning tasks.

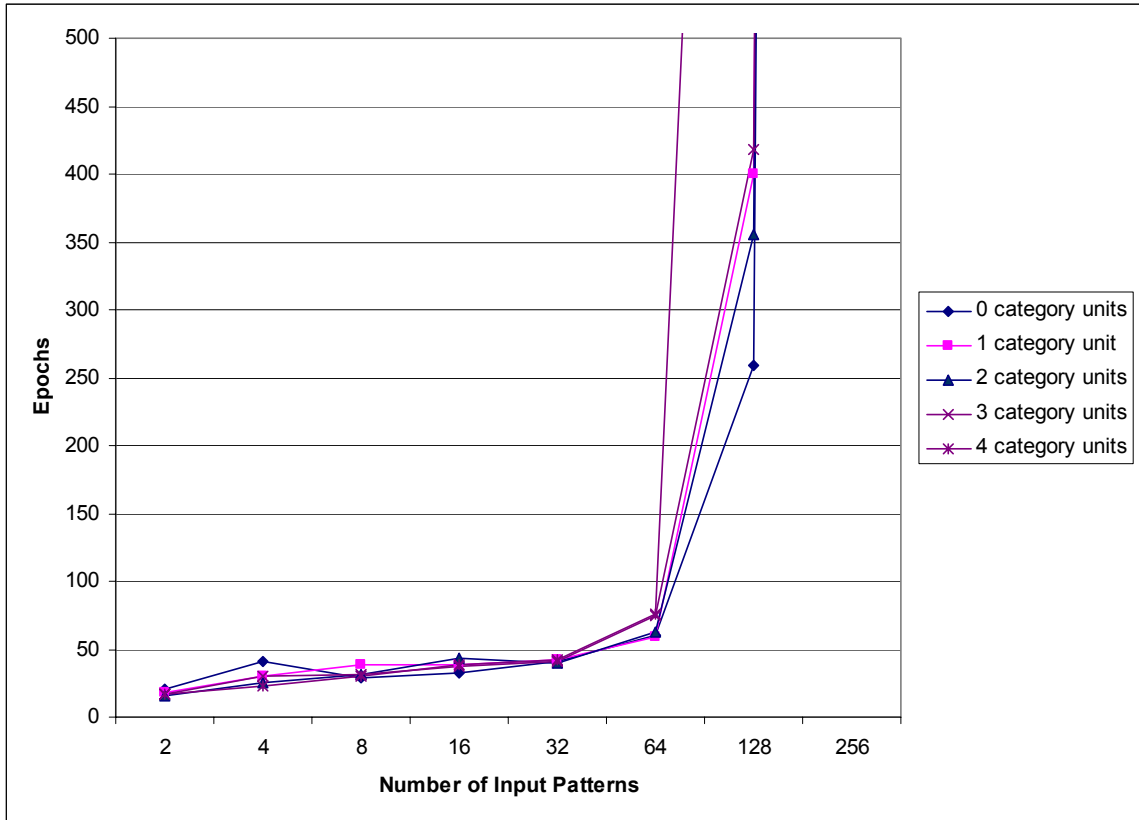


Figure 4.1 Amount of Initial Training (Step B) Required for All Architectures (Case α)

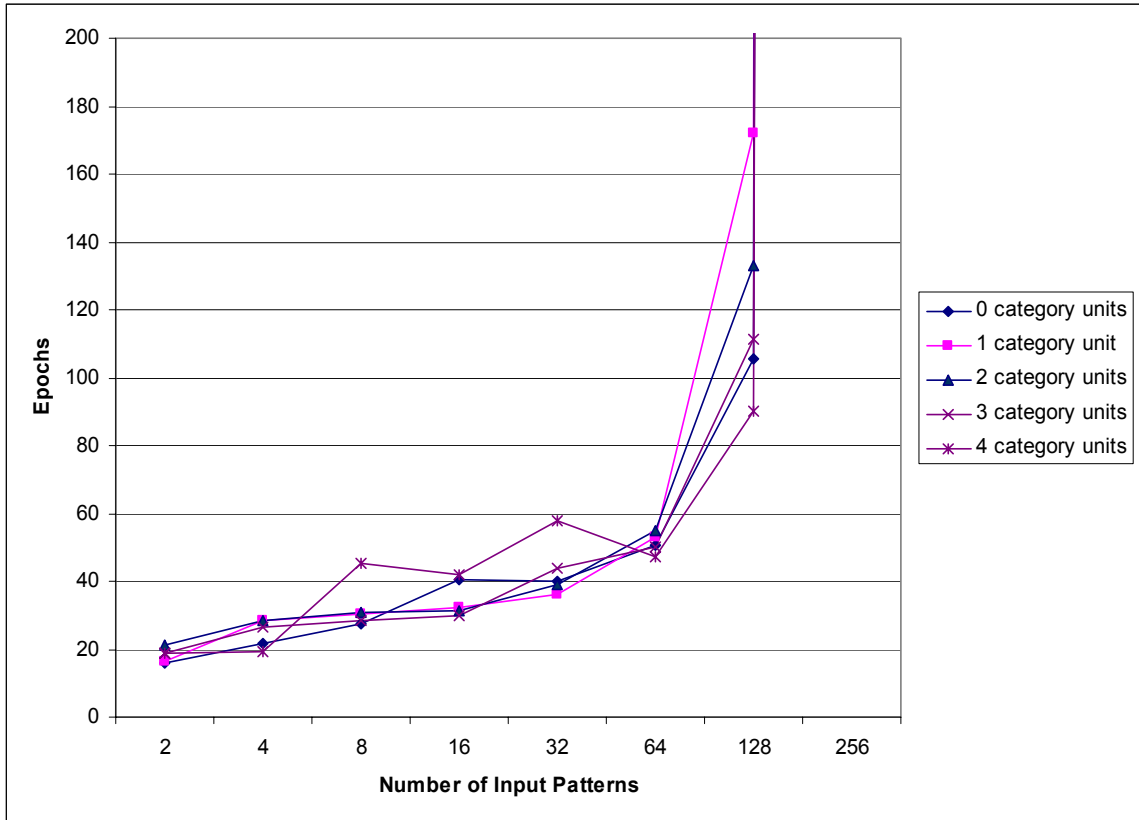


Figure 4.2 Amount of Initial Training (Step B) Required for All Architectures (Case β)

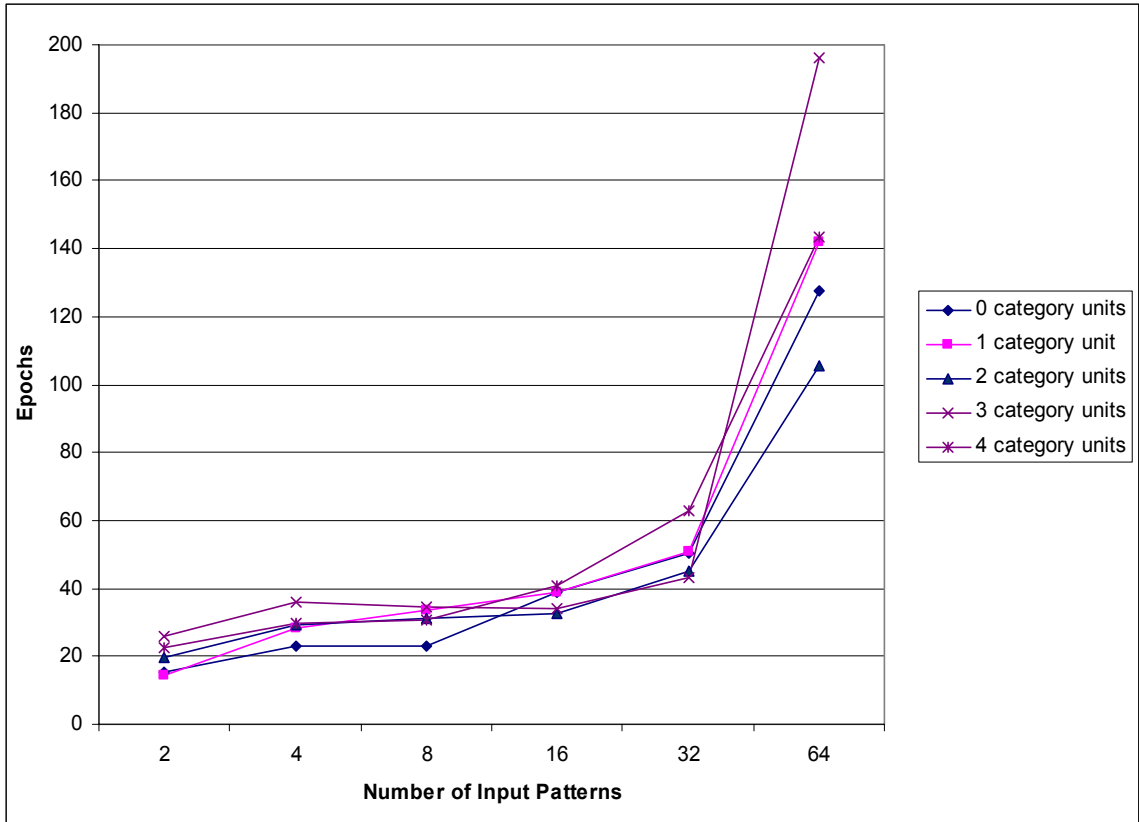


Figure 4.3 Amount of Subsequent Training (Step D) Required for All Architectures (Case α)

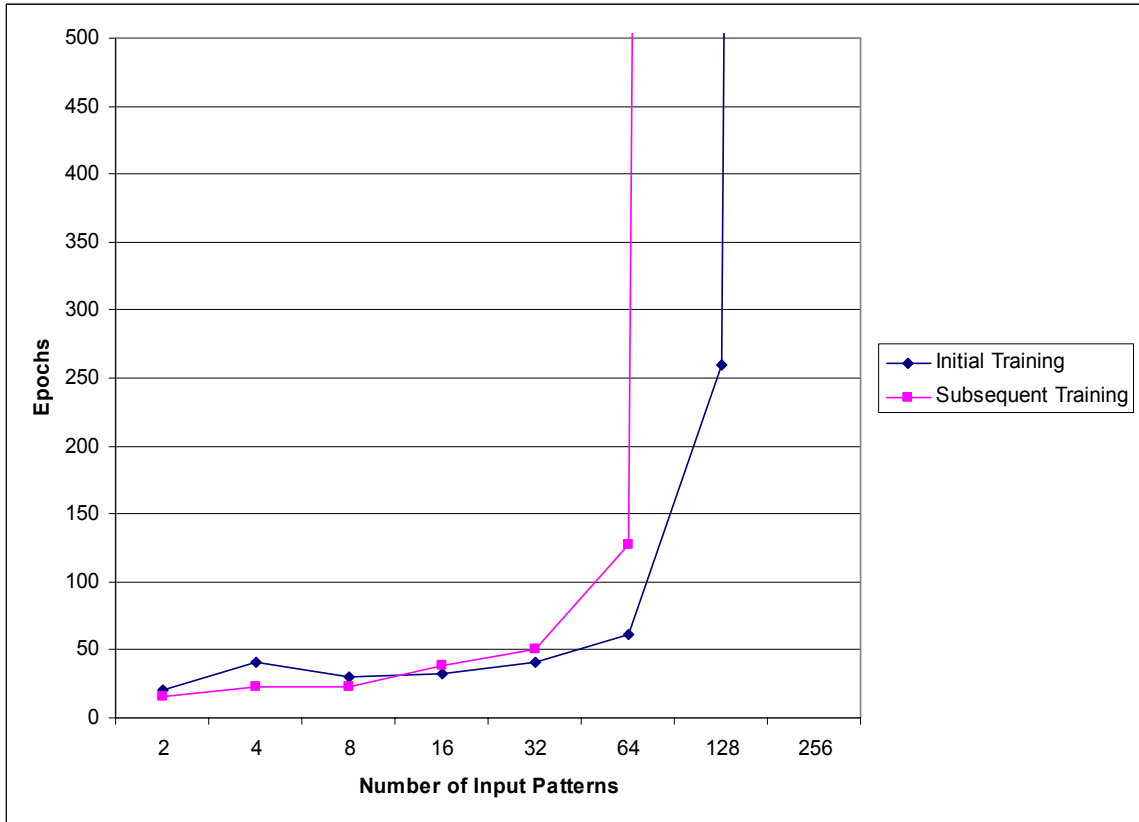


Figure 4.4 Initial (Step B) and Subsequent (Step D) Training Time for Zero Category Units Architecture (Case α)

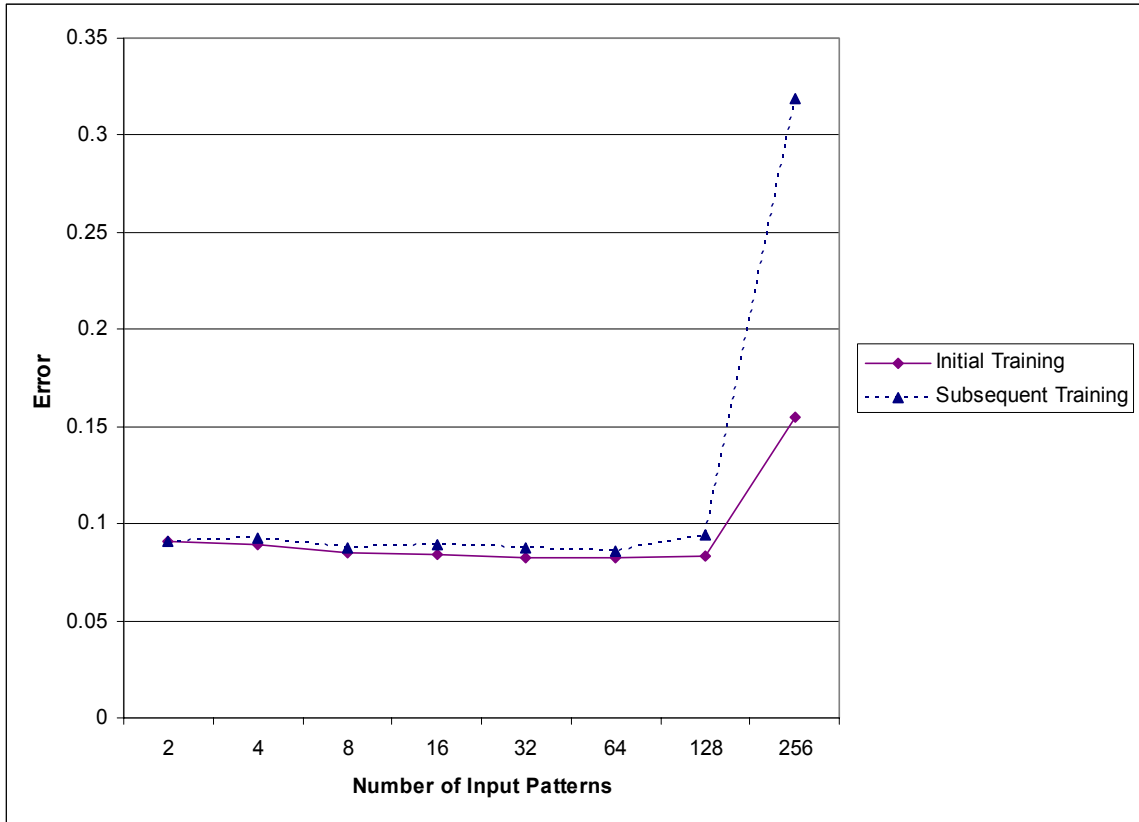


Figure 4.5 Initial (Step B) and Subsequent (Step D) Training ϵ for Zero Category Units Architecture

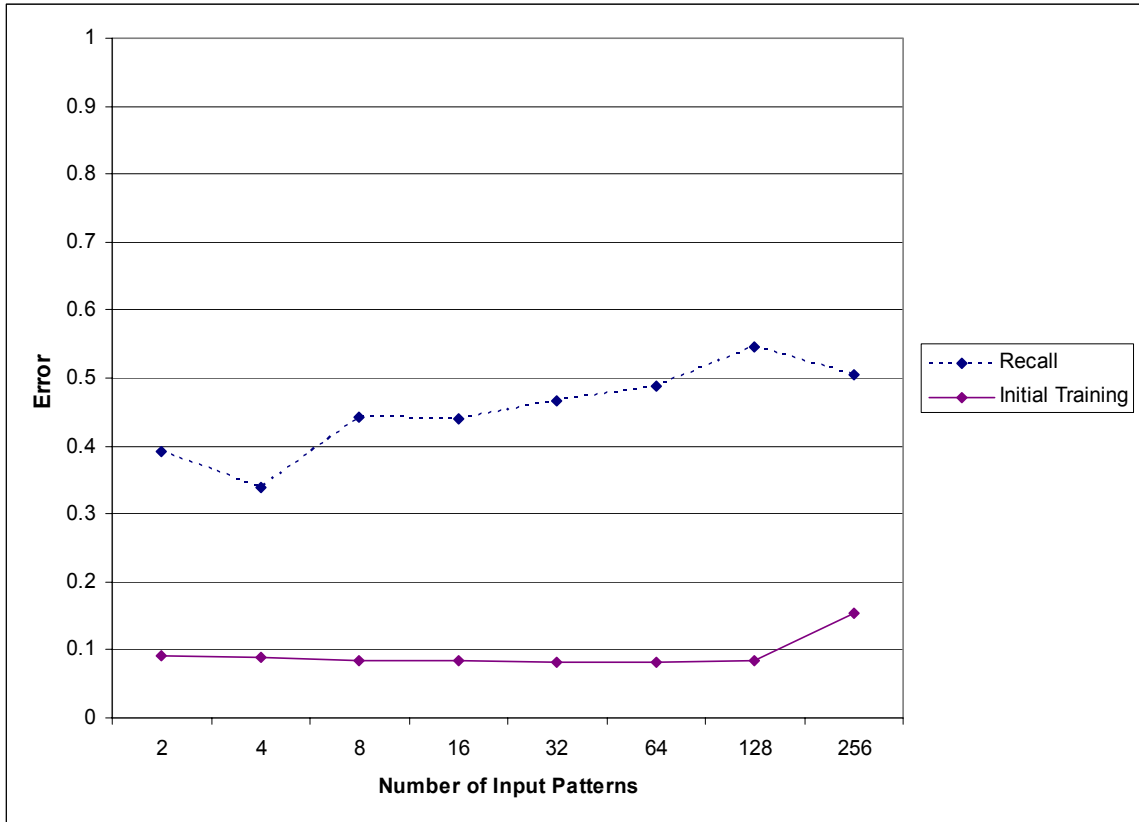


Figure 4.6 Initial Training (Step B) and Recall (Step E) ε for Zero Category Units Architecture (Case α)

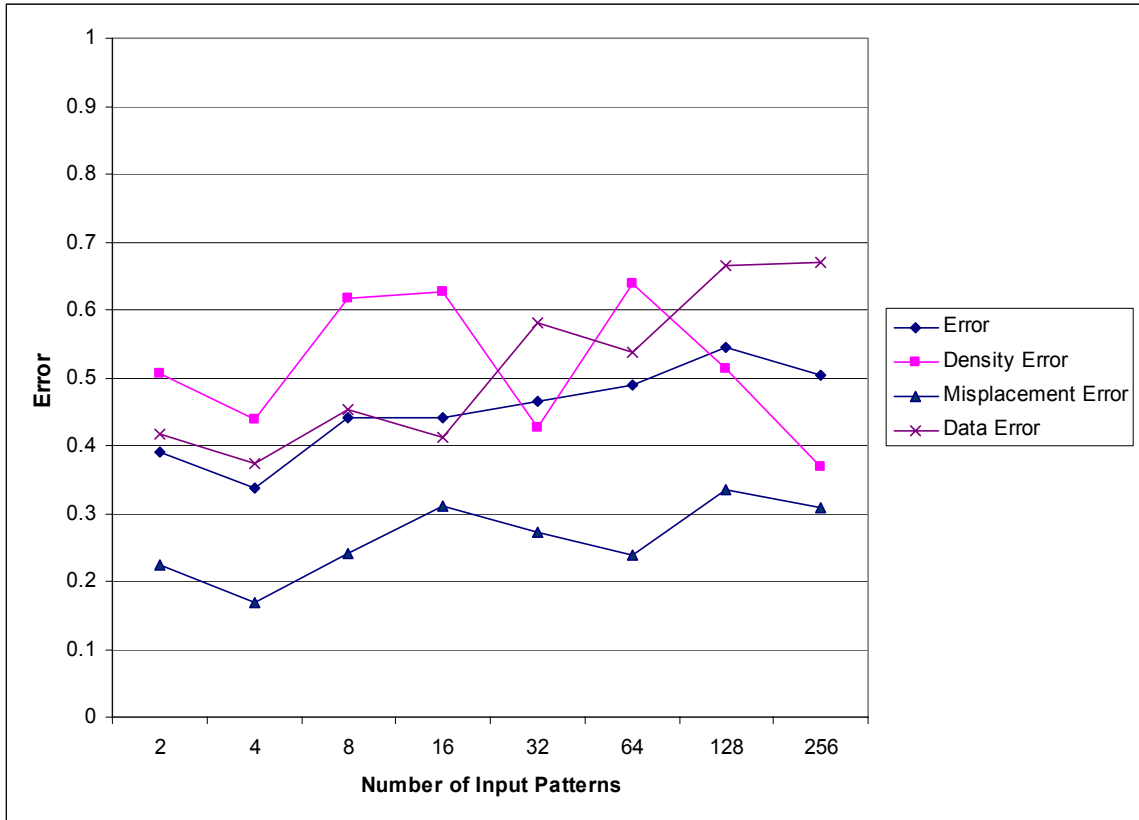


Figure 4.7 Recall (Step E) ϵ for Zero Category Units Architecture (Case α)

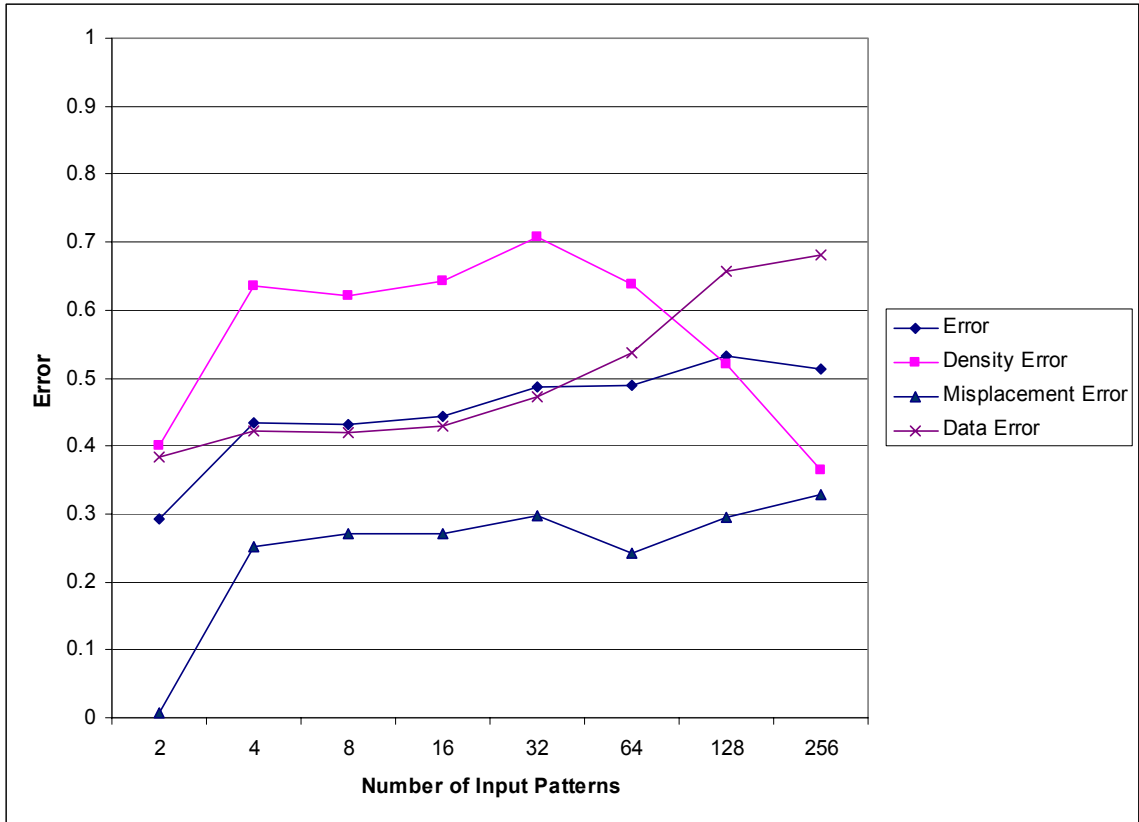


Figure 4.8 Recall (Step E) ϵ for One Category Unit Architecture (Case α)

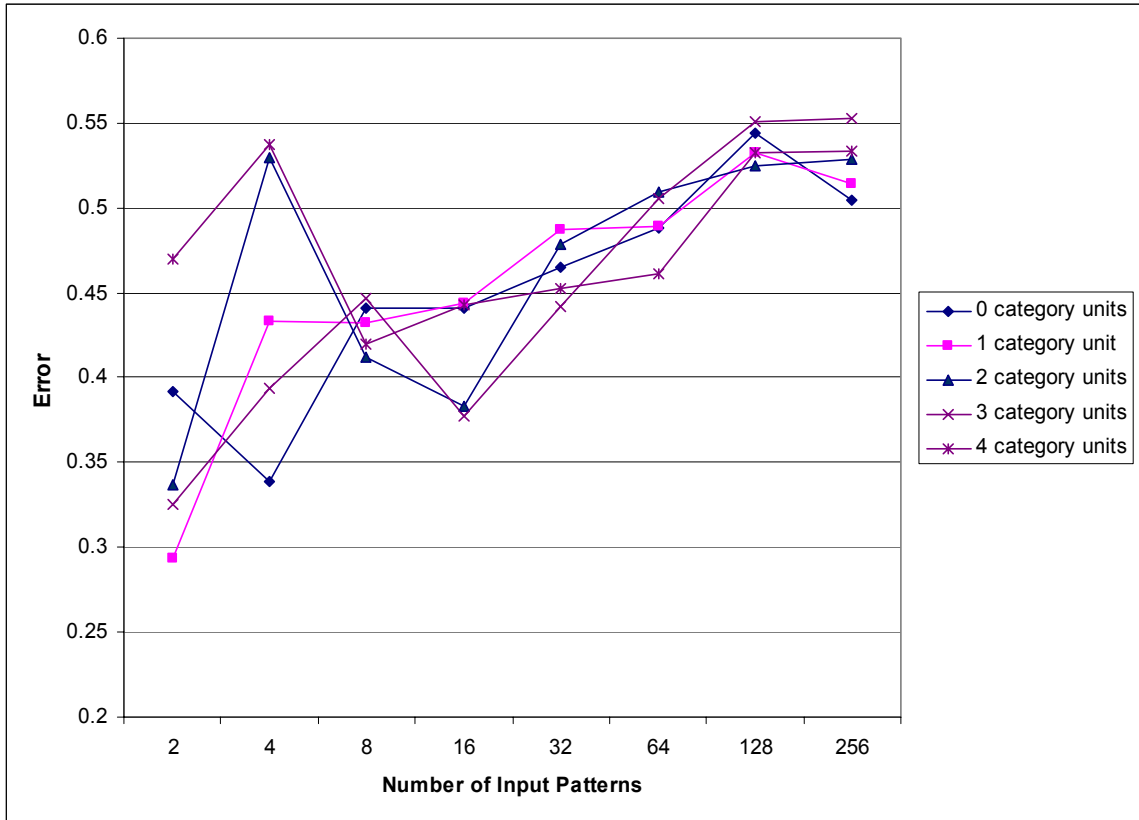


Figure 4.9 Recall (Step E) ϵ for All Architectures (Case α)

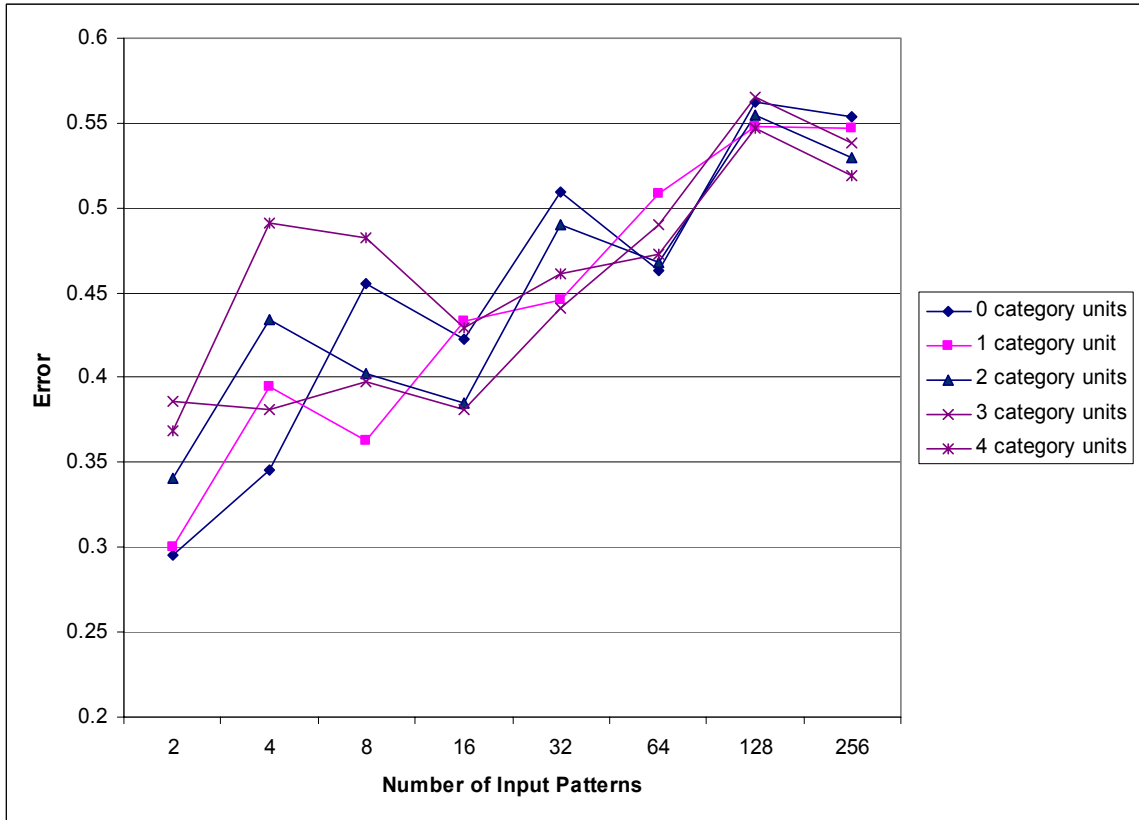


Figure 4.10 Recall (Step E) ϵ for All Architectures (Case β)

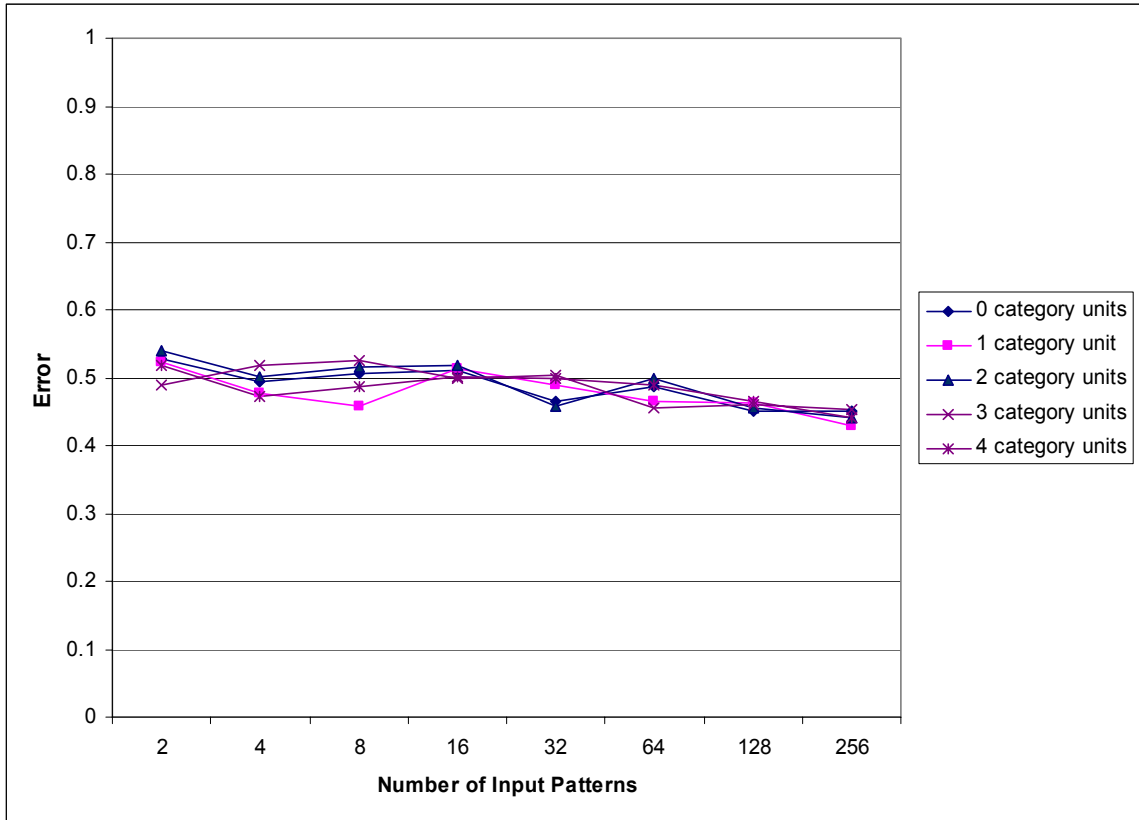


Figure 4.11 Initial Generalization Test (Step C) ε for All Architectures (Case α)

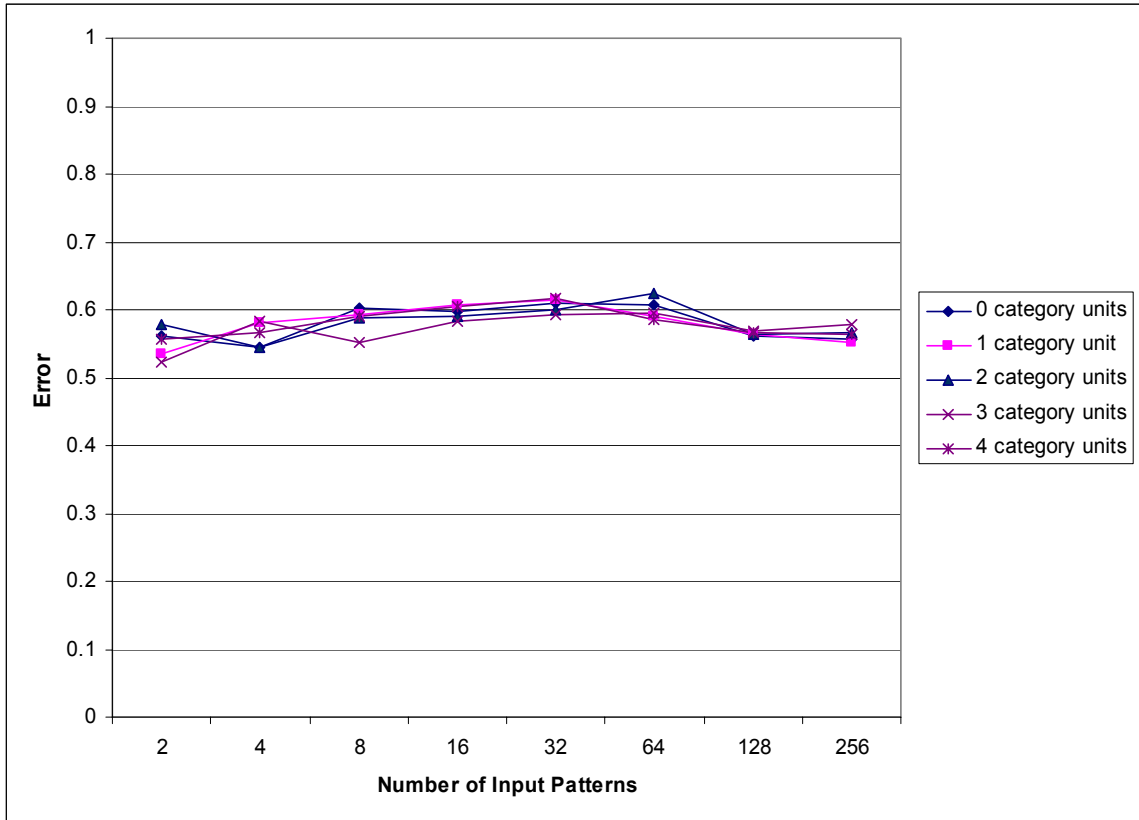


Figure 4.12 Recall Generalization Test (Step F) ε for All Architectures (Case α)

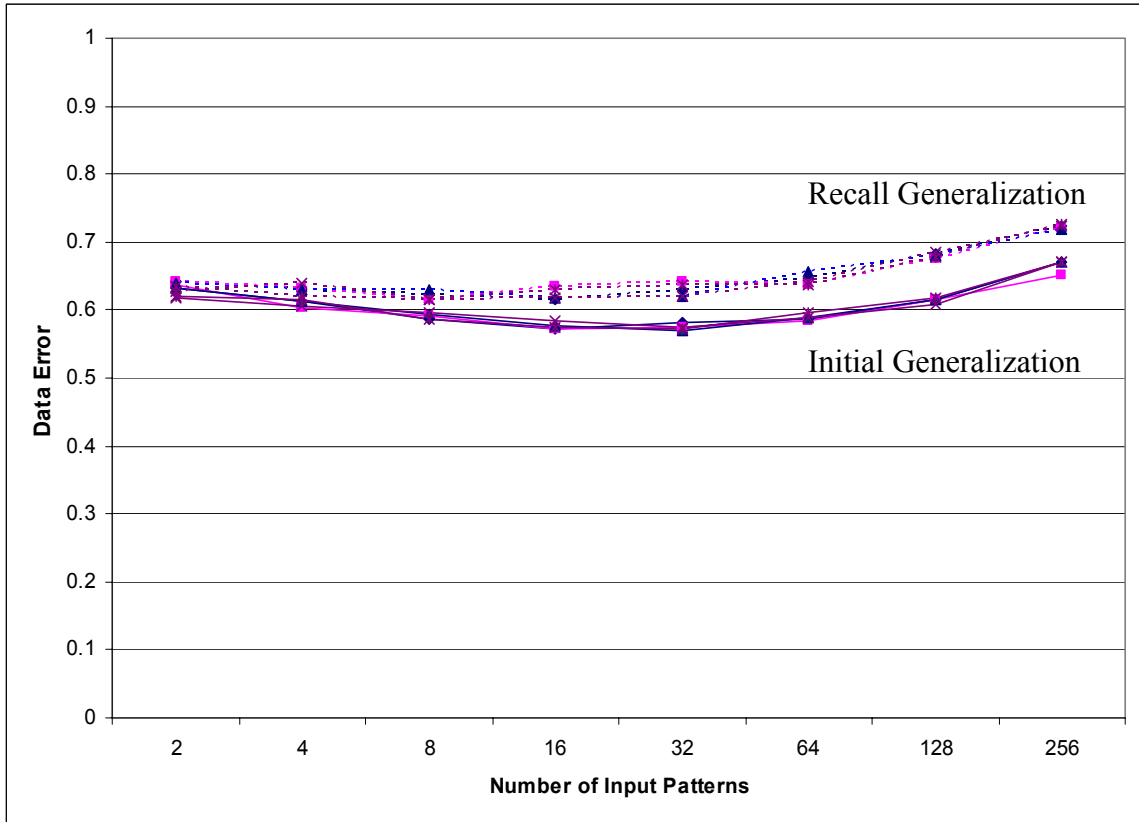


Figure 4.13 Initial (Step C) and Recall (Step F) Generalization Test ε_{data} for All Architectures (Case α)

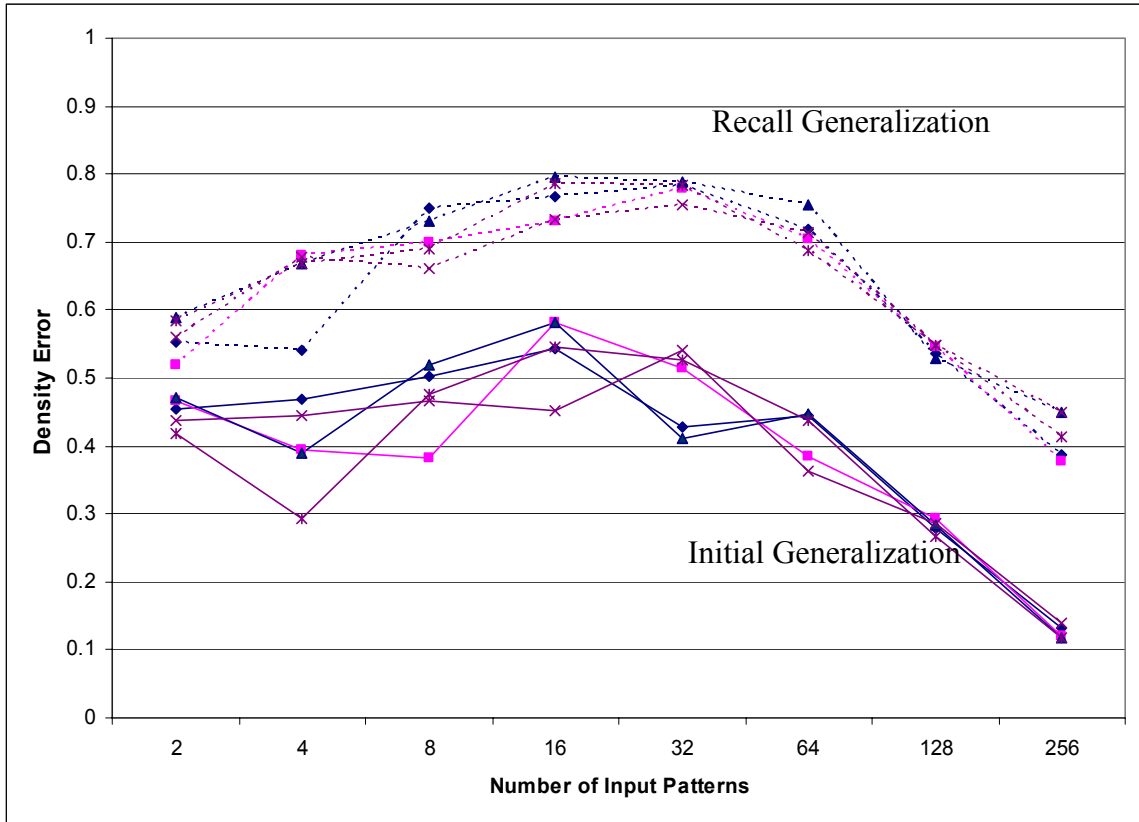


Figure 4.14 Initial (Step C) and Recall (Step F) Generalization Test $\varepsilon_{density}$ for All Architectures (Case α)

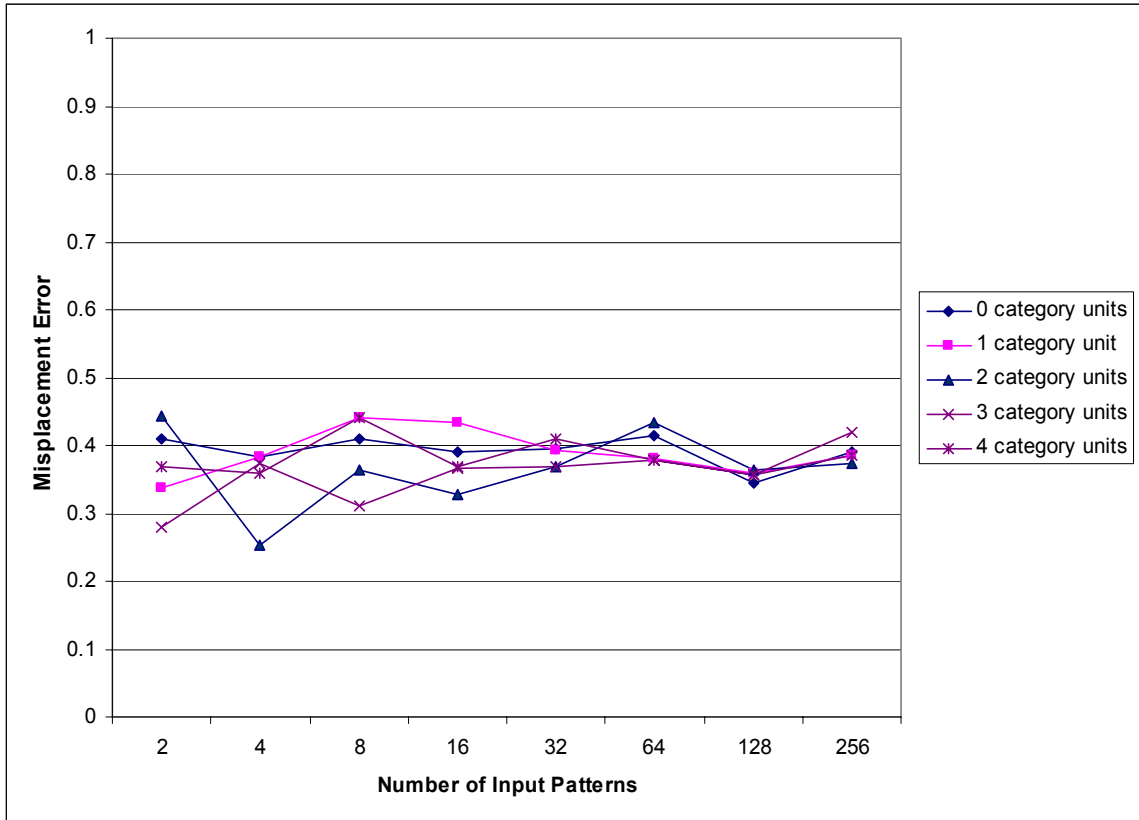


Figure 4.15 Recall Generalization Test (Step F) $\epsilon_{misplacement}$ for All Architectures
(Case α)

Chapter 5. Discussion

5.1 Introduction

We can view many types of learning as association learning. For example, word learning may be considered association learning, because in some cases an audio pattern (i.e., a word sound) is associated with a visual pattern (i.e., an image of the word's referent). People often make use of category information when learning associations. For example, we need to distinguish (i.e., categorize) the sound of words, or human speech, from other sounds (e.g., doors slamming) and we also need to utilize this distinction when associating these sounds with other information.

In this thesis we explored how category information affects association learning in computational models of neural networks. Computational models may help us in understanding association learning and categorization. We hypothesized that category information may assist association learning in neural network models. We analyzed how providing category information affects the performance of association learning using criteria of training time, generalization error and interference. Interference is said to occur when the learning of one task changes the performance of the model or system on another learned task. We hypothesized that providing explicit category information as inputs to a neural network may help it to better distinguish between categories of input patterns and leave a larger scope of possible weight adjustments to capture other properties of input patterns. This may also lead to reduction in interference between associations from different categories.

5.2 Results

In order to evaluate our hypotheses we designed and performed three experiments. In Experiment 1 we assessed the training and generalization performance of association learning models without explicit category information (Model₁, see Figure 2.1, page 20) and association learning models with explicit category information (Model₂, see Figure 2.4, page 24). The learning models were instantiated as backpropagation feedforward neural networks. The data sets that were used in Experiment 1 contained pattern₁, pattern₂ pairs from two categories. Patterns were eleven bits and parity was used to assign categories. Category units, where applicable, were either all 1's or all 0's. Networks were trained in a simultaneous learning task which involved training associations from all categories in a single training set.

In Experiment 1 we found that Model₂ architectures always had shorter training time and lower generalization error than Model₁. Model₂ architectures showed some variation in generalization error with small training data sets and reached 100% accuracy with 32 and more training patterns. Model₁ had relatively large variation in the number of correct patterns in generalization tests and showed an increasing trend in the number of correct patterns in generalization tests with increases in the number of training patterns. Improvements in Model₂ performance compared to Model₁ performance were due to the use of category units as this was the only difference between Model₁ and Model₂.

In Experiment 2A we assessed the effects of category information on association learning when the data was more complex than in Experiment 1. We used more complex data in Experiment 2A because real world associations may be more complex than the

data used in Experiment 1. The data sets used in Experiment 2 provided relatively explicit category information, but yet had a relatively complex function mapping input to output. Data sets were constructed using the schema in Figure 3.1 (page 59). Each pattern had 80 random data bits and a cue marking its category at an arbitrary location among the data bits. A cue was a contiguous sequence of 20 bits of the same value. Each data set contained equal numbers of pattern₁, pattern₂ pairs from the two categories presented in interspersed random order (i.e., a simultaneous learning task was used as in Experiment 1).

The results of Experiment 2A showed that Model₂ architectures in general had somewhat smaller per sample error in generalization than Model₁ architectures, while training time was approximately the same for both models. Improvements in per sample error were due to improvements in cue density error and data error, while the other error term, cue misplacement error, was approximately the same for all architectures. Results of Experiment 2A showed that additional category information may still be beneficial to association learning even when category information is a relatively explicit part of the input and the input to output mapping is relatively complex.

In the closing Experiment (2B), we assessed the effects of category information on association learning in a *sequential* learning task. In this sequential learning task the networks were first trained on examples from one category and then received training on examples from the other category. A typical outcome of sequential learning tasks for connectionist models is subsequent lower performance on the initial task. The loss of performance is often referred to as catastrophic interference due to the relatively severe

degradation in performance. Experiment 2B reproduced Experiment 2A, but this time with a sequential learning task. This allowed us to not only assess the effects of category information on training and generalization but also assess the effects of interference on association learning models. The data sets in Experiment 2B were the same as in Experiment 2A, but we presented $pattern_1$, $pattern_2$ pairs of one category as initial training, and then trained pairs of the other category.

The main finding of Experiment 2B was that category units did not help to reduce interference in the sequential learning task, and in fact, in some cases (e.g., the four category units architecture), category units actually *increased* interference, with data constructed using the schema in Figure 3.1 (page 59). Recall error was higher than the initial training error in both $Model_1$ and $Model_2$. The increase of recall error compared to the initial training error arose from increases of cue density error and data error. Increased cue density error showed that the neural networks were not able to effectively use category information (both from cues and from category units) to distinguish categories of patterns. Higher levels of data error in recall tests compared to initial training were due to a relatively large number of unique data bit sequences. Other results of Experiment 2B were that $Model_1$ and $Model_2$ architectures showed approximately the same performance in terms of training time and generalization.

In this thesis we found that category units may reduce generalization error and training time of association learning models in simultaneous learning tasks. Category units may improve association learning even when the input already contains relatively explicit category information. For example, there was higher generalization in $Model_2$

compared to Model₁ in Experiment 2A. However, when we changed the task in Experiment 2A to a sequential task (Experiment 2B), we found that category units in some cases actually increased interference.

5.3 Future Work

In our study we assumed that category information was available and we did not consider the resources and mechanisms required to acquire this category information (e.g., training time). Also we assumed 100% accuracy of category information. It seems unlikely that category information in all real world applications will have 100% accuracy. Studying the effects of category information on association learning when category information is not 100% accurate should also be performed. In some cases, category information may be correct, but somewhat ambiguous. For instance, if someone talks and there is a loud background noise, sounds could be misclassified as speech or as a non-speech sound. Additionally, it is not clear what effect category information will have on association learning in the case when the same data belongs to multiple categories.

Category units in our models provided extra information about data patterns to association learning models. In a more general sense, category units provided information about a property of data patterns (i.e., that the data pattern was from category₁ or category₂). Different from this, some data have multiple properties and we therefore may want to assign categories for each of the data pattern's properties. For instance, data representing a visual image of a dog may be assigned categories "animate" and "black." In this case, providing category information about both "animate" and "black" properties of data to association learning may improve association learning performance. Studying

the effects of providing categories for multiple properties of data would be an interesting direction of assessing category information effects on association learning models.

In a sequential learning task category units did not help to reduce interference in the association learning models (Experiment 2B). This result seems to be due to the distributed representation of knowledge in the neural networks used. Backpropagation neural networks, as well as some other multilayer feedforward neural networks distribute acquired knowledge among all weights. This distributed knowledge representation is one of the reasons that catastrophic interference occurs (e.g., French, 1999). A typical outcome of subsequent training is modification of all a neural network's weights and this modification erases knowledge about the initial training set. French (1999) reviewed some techniques for overcoming interference in neural networks. These techniques included using localized knowledge representations where new training sets are represented by some weights rather than by all weights (e.g., ART neural networks) and using dual network architectures that enabled rehearsal of previously learned information. Ans and Rousset (2000) used dual network architectures for overcoming interference. In this case, the task of one network is to learn input and output associations (i.e., similar to short term memory) simultaneously with generated input and output patterns from the second network. The task of the second network was to approximate all training patterns (i.e., similar to long term memory) and to generate pseudo patterns used for rehearsal (i.e., refreshing) in the training of the first neural network. The *self-refreshing memory* model (Ans & Rousset, 2000) enables overcoming catastrophic interference. In the reported experiments it seems important that Ans and Rousset used two bits for

distinguishing initial training and subsequent training data sets. It may be the case that the *combination* of category information and specific dual neural network architecture enabled relief from catastrophic interference in the self-refreshing memory model. Replicating the self-refreshing neural network architecture and assessing interference when no category information is provided may shed more light on the issue of the effects of category information on interference in association learning models.

References

Ans, B. & Rousset, S. (2000). Neural networks with a self-refreshing memory: Knowledge transfer in sequential learning tasks without catastrophic forgetting. *Connection Science*, 12, 1-19.

Atkinson, R. & Shiffrin, R. (1968). Human memory: A proposed system and its control processes. In Spence, K. & Spence, J. (Eds.), *The Psychology of Learning and Motivation*, 2, 90-195.

Carpenter, G.A. & Grossberg, S. (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics: Special Issue on Neural Networks*, 26, 4919-4930.

Eysenck, M.W. & Keane, M.T. (1995). Prototype Theories. In *Cognitive Psychology. A Student's Handbook*. Lafayette, CA: Lawrence Erlbaum Associates.

Fausett, L.V. (1994). *Fundamentals of neural networks: architectures, algorithms, and applications*. Upper Saddle River, NJ: Prentice-Hall.

French, R.M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3, 128-135.

Hetherington, P. & Seidenberg, M. (1989). Is there "catastrophic interference" in connectionist networks? *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, 26-33, Hillsdale, NJ: LEA

Hollich, G. J., Hirsh-Pasek, K. & Golinkoff, R. M. (2000). Breaking the language barrier: An emergentist coalition model for the origins of language learning. *Monographs of the Society for Research in Child Development*, Serial No. 262, Vol. 65, No. 3.

Kohonen, T.K. (1988). *Self-organization and associative memory* (2nd ed.). Springer-Verlag, Berlin: Heidelberg.

McClelland, J.L., McNaughton B.L. & O'Reilly R.C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102, 419-457.

Mitchell, T.M. (1997). Ch. 4 Artificial neural networks. *Machine Learning*. Portland, OR: McGraw-Hill.

Neath, I. & Surprenant, A. M. (2003). *Human Memory*, Second Edition. Belmont, CA: Wadsworth.

Peterson, R. C. (2001). *An Integrated Model of Context and Motor Learning for Autonomous Agents* (Report 2001.14). United Kingdom: University of Leeds, School of Computing.

Peterson, R. C. (2002). A context-based architecture for general problem solving. *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*, 107-117. Cambridge, MA: MIT Press.

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 97, 285-308.

- Roberts, W.A. (1981). Retroactive inhibition in rat spatial memory. *Animal Learning and Behavior*, 9, 566-574
- Roy, D. (2000). Grounded speech communication. *Proceedings of the International Conference on Spoken Language Processing*.
- Roy, D., & Pentland, A. (1998) Learning words from natural audio-visual input. *Proceedings of the International Conference on Spoken Language Processing*, 4, 1279 – 1284.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533-536.
- Rundus, D. & Atkinson, R. C. (1970). Rehearsal process in free recall: A procedure for direct observation. *Journal of Verbal Learning and Verbal Behavior*, 9, 99-105.
- Shutts, K.B. & Markson, L. (2003). Category-specific learning and generalization in infancy. *Proceedings of the Society for Research in Child Development (SRCD)*.
- Sporns, O. (2000). Modeling development and learning in autonomous devices. *Proceedings of the First International Conference on Development and Learning*.
- Weng, J. & Hwang, W. S. (2000). An incremental learning algorithm with automatically derived discrimination features. *Proceedings of Asian Conference on Computer Vision*, Taipei, Taiwan, 426-431.
- Westermann, G. (2001). A model of perceptual change by domain integration. *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, 1100-1105. Hillsdale, NJ: Erlbaum.