The Acquisition of Intellectual Expertise: A Computational Model

Lisa C. Kaczmarczyk
The University of Texas at Austin
lisak@cs.utexas.edu
PhD Proposal

May 15, 2003

Abstract

Intellectual expertise is knowledge and ability that a person has that allows them to solve extremely complex problems. It is important to understand how people become experts so that we can improve educational strategies, and help learners achieve their full academic potential. Unfortunately, the process of acquiring intellectual expertise is not well understood. The goal of this research is to build and test a computational theory of this process. My approach is to train artificial neural networks (ANNs) as a model of expert human learning. ANNs address many of the difficulties found in trying to study expertise in humans; they have already been successful in modeling other types of human learning. In completed work, I have built a first version of the model and used it to confirm two hypotheses: (1) An artificial neural network can be used as a model to investigate how people learn under different training scenarios. (2) Different methods for delivering the training material result in different final performance, and best performance is achieved by incrementally increasing the complexity of the material. I propose to complete the dissertation by testing a third hypothesis: Different delivery methods result in different internal conceptual representations and conceptual development, which in turn is responsible for the different performance. The results suggest that psychological learning theory should be considered when designing instructional strategies. My research has shown that, in contrast with symbolic models of cognition, connectionist models are able to model the human learning process, and provide practical insights into how people should be trained.

1 Introduction

An intellectual expert has achieved a level of cognitive development in which she or he can rapidly grasp subtleties of complex problems, and produce very high quality solutions. As the meanings of ”rapid” and ”very high quality” are influenced by societal values, there is no measurable level at which we can officially declare someone to have achieved the status of ”expert”. On the other hand, when deciding if a person is an expert in a given subject, it is often relatively easy to reach consensus.

A goal of formal education is to help students achieve an expert level of understanding in their chosen field. It is important to understand the nature of expertise so that we can improve educational

strategies. As a result of many research studies about expertise, we know a lot about the characteristics of experts. However, there is a lot we don't understand about how to become an expert. It is not easy to create experts, whether human or computational. The learning process is complex and human studies are difficult. Understanding how to acquire intellectual expertise has proven elusive for educators, psychologists and students alike.

A primary goal of this study is to increase understanding of the process by which humans become intellectual experts, in particular: how can people develop the ability to identify solution strategies just by looking at them? The second main goal is to understand this process in the context of formal instruction; specifically, how does the strategy by which material is delivered to the learner affect their learning and conceptual development? I am testing the hypothesis that the acquisition of intellectual expertise takes place within the human mind via a statistical learning process that can be understood using a distributed computational model. I propose to show that artificial neural networks (ANNs) can provide valuable insight into delivery methods (the presentation of material), insight not easily obtainable through conventional studies.

In this document, I present results from a series of experiments examining how different delivery methods influence learning and conceptual development. These experiments use a real-world adult educational problem: the ability to identify correct solution strategies for calculus integration problems.

The work completed so far has focused on two hypotheses. Hypothesis 1: An artificial neural network can be used as a model to investigate how people learn under different training scenarios. The main results include: (1) errors are higher on final exams when problem types are learned in isolation; (2) cramming just prior to taking final exams is associated with poor retention of material. Hypothesis 2: Different delivery methods result in different overall performance. The main results include: different delivery strategies affect learning in different ways: (1) traditional sequential delivery methods inhibit learning and retention; (2) integrated delivery methods increase learning and retention; (3) the best performance comes from delivery methods that incrementally increase the complexity of material.

For the completion of my dissertation I propose to test Hypothesis 3: Different delivery methods will result in different internal conceptual representations and conceptual development. I will test this hypothesis by analyzing the conceptual development that took place during each type of completed experiment. I will analyze this data across experimental types and in the context of what is known about human conceptual development.

The remainder of this document is structured as follows. The Background section defines intellectual expertise and makes the case for using connectionist computational models to study cognition. The Background also introduces and justifies my choice of calculus integration as a domain topic for experimentation. The Model section introduces the ANN model for how intellectual expertise can be acquired. The Experiments sections describe three sets of completed experiments, which correspond to three delivery and assessment strategies for academic course material. The Proposed Work section describes how I intend to continue my research to understand the conceptual basis of the observed results. In the final two sections I provide a psychological learning theory perspective on my results, and contrast my approach with the symbolic approach to computationally modeling expertise.

2 Background and Previous Work

The purpose of this section is to provide the psychological, computational and educational context for understanding the rest of this document. I begin in section 2.1 with a review of what is known about the attributes and behaviors of experts. In Section 2.2 I discuss the difficulties with studying the acquisition of intellectual expertise in humans, and review connectionist models as a way to address these problems. In Section 2.3 I introduce calculus learning as a domain for studying the acquisition of intellectual expertise. First I discuss national concerns with mathematics education, and then I review studies showing that calculus is a cognitively difficult subject for learners.

2.1 What is Intellectual Expertise?

Because "expert" is a commonly used, yet loosely understood term, it is reasonable to begin with a dictionary definition. The American Heritage Dictionary defines an expert as "A person with a high degree of skill in or knowledge of a certain subject" and expertise as "Skill or knowledge in a particular area"(Editor 2000). Unfortunately, and as many educators would readily admit, these definitions are not very helpful as guides for instruction. They do not capture the full depth and complexity of intellectual expertise.

Studies of human expertise and understanding have revealed key information about experts. The first set of findings serves to define the attributes and behavior of experts. In seminal studies, DeGroot (1966) and Chase and Simon (1973) showed that experts possess large domain knowledge and an advanced ability to see patterns. It is this combination that underpins the ability to think critically and solve problems. We also know, from another classic study, that experts and novices categorize problems differently, and that this categorization takes place before the subject attempts to solve the problem (Chi, Feltovich, and Glaser 1981). The same study claimed that the schema containing the categorization includes potential solution methods. Other studies have confirmed the ability of experts to categorize problems without solving them (Hinsley, Hayes, and Simon 1977) (Robinson and Hayes 1978). Finally, there is also strong evidence that routine problems are solved not by intense calculating but rather by recognizing a type of problem (classifying) and then using the stored knowledge about how to solve problems of that type (Reiman & Chi '89 referenced in (Ross and Spalding 1991).

Most studies of expertise have focused on what an expert knows, rather than the process by which she or he attained expertise (Duffy, Lowyck, and Jonassen 1993). As a result, we know a lot less about this learning process than we do about expertise itself. The second, much smaller, set of findings about experts describes the learning process humans go through to achieve levels of expertise. Nevertheless, we know that understanding is closely connected to the ability of a learner to correctly generalize and transfer concepts (Bransford, Brown, and Cocking 2000). Unfortunately, although experts demonstrably "understand" in a manner that novices do not, it is still unclear just what "understanding" means. There is substantial evidence that deep understanding involves primarily the restructuring of existing conceptions of reality as opposed to adding new ones (Fosnot 1996). Strong evidence supports the claim that expert behavior does not simply follow a script: the greatest expertise is the result of long-term practice (Hayes 1989) that is consciously goal directed, self-monitoring, and self-adjusting within the setting of each particular problem task (Garner 1990). In addition, many studies have shown that meta-cognition (self-appraisal and self-management of cognition) is critical for successful academic learning (literature surveyed and discussed in Paris and Winograd (1990)). Since we know that experts categorize extremely well, it is possible that catego-

rization ability and goal-directed meta-cognition enhance one another. When categorization ability and goal-directed meta-cognition merge, intuition may be the result: there is strong evidence that experts rely upon their accurate intuition and a holistic recognition of appropriate actions (Dreyfus and Dreyfus 1986).

2.2 The Case for Connectionist Models of Cognition

This section describes how connectionist computational models can provide insight into how people become experts. Section 2.2.1 reviews common approaches to studying cognition using human subjects, and the limitations of these approaches. Section 2.2.2 discusses the advantages of using computational models to study cognition, and then reviews previous connectionist research into learning.

2.2.1 The Difficulty with Studying Expertise Acquisition in Humans

Research on human cognition takes place within many disciplines and at many levels, ranging from neuro-biological analysis to field observations. Traditionally, many such studies have focused on tightly controlled and isolated phenomena. This control tries to eliminate irrelevant data and provide generalizable results. Unfortunately, it is difficult to apply these results to realistic learning situations, which tend to be uncontrolled and strongly influenced by the environment. Adult human learners, for example, rarely study under tightly controlled conditions. Their intellectual tasks are often extremely complex. As a result, it is unclear how to apply experimental results to academic teaching and learning: we cannot simply assume that results transfer to more complex problems or a different population. This concern is particularly relevant for the study of intellectual expertise. Since expertise is all about managing complexity, complex problems need to be the focus of study.

Another experimental approach is that taken by many recent educational studies, which borrow qualitative research methodologies from the social sciences. Qualitative methodologies traditionally focus in great detail upon a small number of human subjects. As a result, they provide a holistic understanding of phenomena. Unfortunately, the small sample size means that results from qualitative studies cannot be generalized beyond the study population.

2.2.2 Connectionist Models of Learning and Cognition

Computational studies can avoid many of these difficulties. They have many well-established advantages over working with live subjects. First, computational models can bridge disciplinary boundaries and provide insight not easily obtainable through conventional studies on humans. For example, unexpected or easily overlooked trends and patterns of behavior may stand out in graphical analyses. Second, successful simulation results make a strong argument for supporting human studies that otherwise might lack political or financial support. Third, simulations and models can minimize the impact of the complexity of phenomena without ignoring it. Fourth, simulations can make it easier to study poorly poorly understood phenomena, such as potentially risky hypotheses, without risking human participants. Finally, simulations can appear to compress time. This allows the researcher to perform longitudinal investigations that would otherwise be impractical or even impossible to do.

Modern studies in neuroscience have demonstrated that the brain has a parallel and highly interconnected structure. We know that during learning neurons are physically rewired, and connection

strengths change (Bransford et al. 2000). This information has inspired the branch of computational cognitive studies known as connectionism (Rosenblatt 1958). Learning is posited to occur via the interaction of many components, which simultaneously constrain one another. Instead of a single processor, there are many processing elements. This architectural interpretation leads to the conclusion that knowledge is implicit in the structure of the device (brain or computer) as opposed to being explicit in a given state (Rumelhart 1998). Changing patterns of connectivity determine what is known and how the system responds to future input.

Connectionist models may provide answers for many classes of cognition problems. The chances for these successes are high because connectionist models have properties that align well with properties of human cognition. For example, the memory behavior of human experts exploits rapid pattern matching and completion, and recognition response (DeGroot 1966), (Chase and Simon 1973). Also, both connectionist models and humans detect correlations between actions and their outcomes, and adapt in response. In addition, connectionist systems allow defining learning procedures that permit the system to adapt to unforeseen stimuli (input). They also exhibit a gradual adaptive ”forgetting” behavior when unused connections gradually lose their influence on decision-making (output). Finally, connectionist networks learn via repeated training, which facilitates the study of the process of learning. The ability to study errors, and corrective adaptations to complex intellectual problems, opens up new and exciting avenues investigating how artificial and biological systems become experts.

There have been some noteworthy successes using ANNs to model human learning. Areas in which connectionist models have already expanded our understanding of cognition include the development of infant vision and perception (Bednar and Miikkulainen 2003), (Chaput and Cohen 2001) language acquisition and comprehension (Elman 1991a), (Miikkulainen 1997), (Rumelhart and McLelland 1987), musical analysis (Todd, 91, as cited in Chen and Miikkulainen (2001); and (Rumelhart 1998)), and memory (Alvarez and Squire 1994), (McClelland, McNaughton, and O’Reilly 1995). A particularly interesting early connectionist model of learning was presented by Viscuso, Anderson, and Spoehr (1989). Their ANN simulated qualitative reasoning while doing multiplication. This study is intriguing for many reasons. First, it successfully modeled how experts estimate correct answers. Second, by analyzing the type and frequency of errors during learning, they showed similarities between their model and humans. Third, they successfully modeled association errors and showed that related data caused confusion, as it does with humans. The types of association errors they saw varied depending upon the order in which problems were learned and how much they were practiced. In summarizing their model, Viscuso et al. correctly pointed out that the most important contribution of their model was the ability to mimic the manner in which experts rely not so much on formal logic and rules but on their ”sense” of what is correct. Since this study, significant advances have been made in technology and understanding of cognition, and it should be possible to give these results another look and to follow them up with new studies.

Another interesting early ANN system learned to perform arbitrarily long addition problems (Cottrell and Tsung 1993). Their model was shown to learn the implicit underlying rule of addition. Cottrell and Tsung also analyzed the internal states of the model at different stages of learning and plotted the principal components of the results. They found that the network state space distinguished between actions needed to do addition: CARRY, WRITE, NEXT. Although giving precedence to the existence of rules in problem-solving, this system showed the flexibility of ANNs for cognitive modeling: the network learned an important concept that it had not been explicitly trained on.

These studies show the wide range of cognitive tasks that can be successfully modeled with

ANNs. They also show the ability of ANNs to successfully model complex learning. In the next section I will explain why another good domain to study with an ANN is calculus learning.

2.3 Why Calculus Learning is a Good Domain to Study

This section provides the context for understanding why calculus is a good domain to use if we want to study the acquisition of intellectual expertise. Calculus, at its most fundamental level, is based upon abstract cognitive concepts. As a result, understanding how people best learn calculus requires understanding the mind. Part of the current educational debates over mathematics and science education, which I discuss in Section 2.3.1, rest upon the reality that we do not understand enough about how the brain produces cognition and conceptual understanding. As I mentioned earlier, neuroscience has made great strides recently, and we know more about the brain than ever before. But neuroscience is no closer to resolving open questions about academic learning than are the educational studies discussed in this section. As a result, we have an opportunity and a need for interdisciplinary researchers to step in and bridge the gap (Bruer 1997), (Selden and Seldon 1997). So Section 2.3.2 lays the groundwork for my cognitive study of calculus learning by discussing why calculus is so hard to learn. Section 2.3.3 reviews previous research in calculus education.

2.3.1 National Concerns with Mathematics Education

In the last decade there has been an increasingly wide-spread national concern with properly educating citizens for the new technological age. A common view is that there is a mismatch between the current educational system and economic need. (For a detailed analysis of this argument, see Marshall and Tucker (1992)). Most would also agree with findings that children and adults do not use strategies necessary to enhance learning (Garner 1990). There is widespread concern among academics that college students do not have critical thinking and generic problem solving skills. Math and science achievement have been at the center of these concerns (for a few examples see (Abudiab 2001), (Medley 2000), and (Moore and Wick 1994)). Large-scale international studies have compared science and mathematics education in the United States with other countries (Atkin 1998), (Stigler and Hiebert 1999). The results show that successful learning is often tied to cultural systems: it isn't possible isolate pieces and transfer them to different environments.

Far from resolving disagreement about which instructional strategies to use, studies such as these have increased the debates. Some people agree that educational change is overdue, but feel that proposed reforms do not go far enough to address individual and cultural needs. Reformers often want education to focus more on empowering the individual learner, and incorporating social interaction into the learning process. Some of these reformers directly challenge institutional power structures as well. For an example, see (Eisenhart, Finkel, and Marion 1996). On the other hand, there are those who believe that reformers are on the wrong track altogether and that a positivist approach to instruction will produce the most intellectually equipped students. Positivism is a doctrine that contends that the only valid basis for human knowledge is what can be directly observed - educators who are positivists often center their teaching and learning around the Scientific Method. For an example of the Positivist argument for education, see (Cromer 1997).

Mathematics is central to the controversies because it is required for success in so many careers. Bruner has even suggested that learning mathematics may be viewed as a microcosm of all intellectual development (Bruner and Kenney 1965). In college, calculus is a gateway course; in order to pursue most studies in the natural sciences and engineering, students must do well in calculus.

Unfortunately, students often have problems with basic calculus concepts. Why is calculus so difficult to learn in the first place? The answer to this question is elusive, but it is important because it impacts decisions about instructional strategy. The next two sections will explore why calculus is difficult to learn.

2.3.2 Calculus Integration as an Ill-Structured Domain

Calculus is difficult to learn because, as with advanced mathematics in general, it belongs to an ill-structured domain. Ill-structured domains have the following two properties: concepts and cases are complex, and not all cases in a concept share the same surface-level structure (Duffy and Jonassen 1992). These properties cause problems for learners who, through well-intentioned attempts to simplify, are taught isolated or superficial concepts. These learners are unable to master complex concepts and transfer knowledge (Duffy and Jonassen 1992). The difficulty appears when the learner encounters advanced material, and discovers that cases nominally of the same type in fact have flexible category boundaries. When these results are viewed along with the evidence confirming that novice learners in general are misled by ambiguity (literature reviewed in (Glaser 1987)), it becomes especially important for delivery strategies not to reinforce simplistic conceptions. If students do not encounter ambiguity until after they have formed problem-solving habits, they will be ill-equipped to handle tasks requiring fine discrimination.

In order to discover what types of problems calculus students are having at my institution, I performed structured interviews with members of the UT mathematics faculty and teaching assistants (TAs) in the fall of 2001. A primary purpose of the interviews was to discover faculty/TA perceptions of a significant student learning problem. Faculty were asked about how students approached calculus integration problems and where they had trouble, and also how they personally approached integration problems. The interviews were part of a study to gain insight into the complex relationship between expert understanding (in this case faculty and TAs) of their own problem-solving process, how they viewed the novices they taught, and the teaching strategies they chose. The interview results fit well with the psychological literature on expert/novice behavior. Novice learners (in this case UT students) are often unable to select the correct integration solution strategy. This fundamental problem arises before they even have a chance to exhibit computational difficulties and prevents many from reaching timely, correct solutions. Conversely, the experts claimed an ability to ”just see” the correct strategy, yet were unable to articulate how they knew. Probing revealed that although there are ”rules of thumb” to assist in strategy selection, they are not comprehensive and do not cover many common scenarios. Experts instead pointed to general patterns and categorization that they have learned to recognize via extensive practice. As would be expected in an ill-structured domain, there is often more than one way to solve an integration problem. The faculty and TAs in my interviews acknowledged this and said that they intuitively recognized each time, which was the best strategy. For example, Integration by Parts can be used to solve the same problems as Usubstitution, but often more easily and efficiently. Also, both Integration by Parts and Usubstitution can be used to solve problems easily solved using Simple Integration. Learners need to acquire this advanced understanding and develop the intuitive ability to just look at a problem and choose the most efficient method. Under pressure, such as on an exam, there is no time to guess and backtrack. If students have only a superficial understanding of problem types, they will likely assume that categories are disjoint. This will lead them to select strategies based upon inappropriate cues (such as surface-level features, or order of placement on an examination), which are likely to fail. Other results supporting my interview data can be found in a study by Burton (1999).

2.3.3 Previous Research in Calculus Learning

Numerous studies in the last two decades have tried, with some success, to gain greater insight into problems learning calculus. Ferrini-Mundy, and Graham surveyed studies from the 1980s and early 1990s (Ferrini-Mundy and Graham 1994), many of which focus on student mis-conceptions of isolated mathematical concepts such as function, limit and derivative. Ferrini-Mundy and Graham also present original results focusing on students receiving ”traditional” instruction and assessment, where traditional is defined as being a large lecture format, without use of technological aids. They concluded, for example, that students show powerful tendencies to call upon familiar examples and frequently-used patterns; traditional means of academic assessment mask rather than reveal the nature of student understanding; traditional language used to explain concepts to students may contribute to their having poor understanding.

Other studies also support the conclusion that traditional teaching methods hinder learning calculus. Selden, Selden, and Mason, conclude that isolated, trivial problems, the norm in many classrooms, inhibit students from acquiring the ability to generalize calculus problem-solving skills (Selden, Selden, and Mason 1994). Similar results are reported by Norman and Prichard (1994). They demonstrate that many learners can not interpret the structure of a problem beyond surface-level symbols. They show that novices have inaccurate intuitions about problems which lead them to attempt incorrect solution strategies (Norman and Prichard 1994). Because they can not see beyond high-level features, they can not develop correct intuitions. On the other hand, successful problem solvers categorize math problems based upon underlying structural similarities and fundamental principles (Silver 1979), (Schoenfeld and Herrmann 1982). These categories are often grouped based upon solution modes, which the experts use to generate a forward working strategy (Owen and Sweller 1989).

Other intriguing data about calculus teaching and learning comes from a recent international study examining the form and style of mathematics and science textbooks. Although the researchers focused on pre-college texts, calculus textbooks were included. This study first pointed to studies showing the substantial impact of the content and structure of textbooks on teacher’s instructional decision-making. They then presented original results showing that textbooks in the United States were uniformly much larger and longer than elsewhere; the textbooks covered far more material, both on a given page and throughout the text, and this material was covered in far less depth than the textbooks from other countries. The researchers went so far as to say that “focus... is not a characteristic of US [math] textbooks overall” (Valverde, Bianchi, Wolfe, Schmidt, W.H., and Hwang 2002). The implication of their results is that these textbooks support the surface level learning that college level calculus researchers have been reporting in their studies.

Other researchers have focused on the role of practice. For example, Resnick and Ford claim that psychological studies have ignored the interaction between instructional methods and understanding (Resnick and Ford 1981). They call for finding out: what regularities are most likely to be noticed, and how does the form in which the initial procedure is taught affect what is noticed? Are there ways of managing how learners practice, to enhance the likelihood that they invent shortcut procedures (their term for what I refer to as intuitive recognition).

All of the studies discussed in this section help to clarify an important research goal: to find the best way(s) to present material so that students will notice subtle interrelationships, and incorporate this information into their problem-solving strategies. One of the first instructional decisions is what order to present the material in, and how to move from one concept to the next. There are many possible orderings of material, and a computational model can be used to explore them. My model,

described in the next section, contributes to acheiving this research goal.

3 Description of the Model

In this section I present a description of the architecture of my ANN model, including how the input data was selected and encoded. The information in this section applies to all three experiment types that I have completed. In sections 3.1 & 3.2 I will describe the architecture of my ANN and describe how how it works. In section 3.3 I will describe the rationale I used to choose the calculus integration examples and to divide them into training and test sets. In section 3.4 I will explain the encoding system I used to convert the calculus problems into a machine readable input vector.

3.1 Architecture and Function of an ANN

My model is an artificial neural network utilizing the backpropagation algorithm created using the LENS network simulator (Rohde v2.3.1). See Figure 1 for a diagram of a typical backpropagation network. Such a network is made up of three layers: input nodes, hidden nodes, and output nodes. In Figure 1 there are 6 input nodes, 3 hidden nodes and 3 output nodes. This is a fully connected network, which means that every input node is connected to every hidden node, and every hidden node is in turn connected to every output node. All of the connections between nodes are weighted (not shown). My model is an enlarged version of this structure, with 55 input nodes and 20 hidden nodes. I will refer to it as a 55-20-3 ANN.

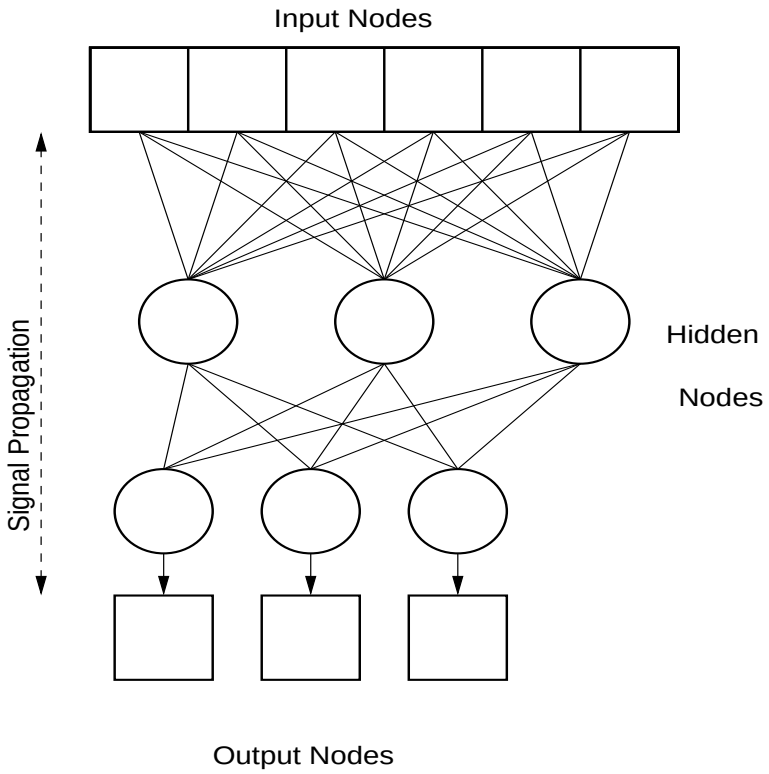

Figure 1: Structure of an Artificial Neural Network.

Although it is beyond the scope of this document to fully explain the operation of an ANN,

it is helpful to include a general description. This description applies to the model used in my experiments. (For a full description of the ANN backpropagation algorithm, there are many sources, such as Bishop (1995), Haykin (1999)).

When ANNs are used for categorization problems, there are two primary operations: training and test. During training, the network is given examples of data one at a time through the input layer. It is asked to decide what output category each problem belongs to. Each node in the output layer represents a category. These training problems have known answers, which are also provided to the network. Without looking at the answer, the network analyzes the problem and makes a guess. The guess is announced by a signal spread across the output nodes. The strength of the signal on each output node represents the strength of the guess (i.e. the aposterior probability) that that particular category is the answer.

The way the input, analysis and guess takes place is commonly spoken of as a signal "propagating forward": from input nodes, through the hidden nodes, to the output nodes. The weights connecting all the nodes influence which output node will receive the strongest signal(s), and become "the guess". At the output layer, an error value is calculated, which represents how close or far the guess was from the correct answer. This error value signal is then "propagated backwards" (hence the name "backpropagation") from the output layer nodes, through the hidden nodes. Along the way, the weights connecting the various nodes are adjusted using an error-correction rule. The hidden nodes together represent the current state of conceptual development of the network. This understanding is spread across the individual nodes; as with human neurons there is no identifiable piece of information contained in any one node. The hidden nodes acquire their values over time, as the network is trained on a given problem. They extract features from the input problems during training. Because there are usually fewer hidden nodes than input nodes, the hidden nodes are forced to create a condensed, complex description of the most salient information that passes through. Thus conceptual understanding, as represented by the hidden nodes, grows and changes with new input.

When training is finished (the timing depends upon the research goals), no more changes are made to the network. During the test phase, the network is shown new problems, which may or may not have known solutions. The network is then "tested" to see how well it performs.

3.2 The Role of the Different Nodes

The nodes in an ANN fulfill different functions. Typically, the input nodes are used to feed data into the network, with each node containing one piece of a larger data item. In my model, an item of data is one calculus integration problem. Each of the 55 input nodes indicate the presence or absence of a visible feature of the integration problem. (In the next section I will describe in detail how these features were determined and coded.) My model has three output nodes, each of which represents one of the acceptable calculus integration solution strategies: Simple Integration, Usubstitution, Integration by Parts. The output nodes in my network report not only the guessed answer for each problem, but how confident the network is in that choice. Each time the network guesses at an answer, it reports values across all three output nodes. The values in the output nodes are normalized so that they sum to 1.0 (100%). The node with the highest value is the preferred guess. For example, if the network reports: 100% 0% 0%, then it is absolutely sure that the problem belongs in the first category. On the other hand, if the network reports: 12%, 85%, 3%, then it is quite confident in the second category, considers the first category possible but unlikely, and the third category extremely unlikely (but not absolutely impossible). If the network reports: 40%,

35%, 25%, it is confused.

3.3 Selection and Design of Training and Test Data

Calculus integration is a complex domain, so care needed to be taken in deciding which problem types to model. It was not computationally possible, or even desirable, to model all calculus integration problems. My goal was to select a subset of problems that would reflect the domain complexity and provide generalizable results. Therefore, in order to balance computational considerations with educational justification, I made the following choices.

There are three types of input data: Simple integration problems (Simple), Usubstitution integration problems (Usub), and Integration by Parts integration problems (Parts). These are the solution strategies most commonly taught to introductory calculus students. A core set of 100 problems came from a variety of college level calculus textbooks (Lang 1986), (Silverman 1985), (Stewart 1995); I generated additional problems using an automated procedure which rearranged variables and operators into legal orderings with the same structure as the core set. Each generated problem was manually checked to make sure that it had a valid syntax before it was included in the data set. All together 957 problem examples were obtained this way. These examples were randomly divided into 10 or more experimental pairings of test and training sets, ensuring that there were never any problems in the test set that were also part of the training set. The reason for multiple pairings is to be able to look for generalizable, statistically reliable patterns of behavior over multiple experimental runs.

Duplicate problems were allowed in the test set because textbooks frequently re-introduce problems with only cosmetic changes, e.g. by altering the value of a constant. Similarly, permutations of multiplication terms were allowed (such as a*b and b*a). The following additional constraints were placed on the input set: all binary arithmetic operators (addition, subtraction, multiplication and division) were included, as was the unary operator for exponentiation (^, e(x)). Trigonometric operators and the natural logarithm (ln(x)) were also included. All these operators allowed the data set to contain a large fraction of possible integration problems.

Some experiments required training sets containing mixes of problem types. The mix of problems in Simple-Usub (SU) training sets was approximately 2:3. The mix of problems in Simple-Usub-Parts (SUP) training sets was approximately 2:3:3. This division corresponds to the greater emphasis that the original textbooks placed upon Usub and Parts problems.

Some of the test sets (which in the experiments are sometimes referred to as ”midterms”) also consisted of mixed problem types, either SU or SUP. In these cases, the test sets were designed such that the more advanced problem type made up proportionally more of the examples in the set. This decision reflects a common instructional practice of including a greater percentage of more recent/complex problems on comprehensive examinations. For this reason also, the exact percentages were not held rigid, but allowed to vary from one experimental run to the next. Test set sizes never exceeded 20% of training set size.

3.4 Data Encoding

Data input to the network is represented using feature coding. Feature coding is a logical choice, given that both novices and experts use the features of a problem to determine which approach to use (Chi et al. 1981). Bishop (1995) discussed techniques for determining which features to include in an input vector, emphasizing that it is important to choose the features carefully. Features influence

the outcome of analysis with a computational model as much as they do with a human. For my experiments, if I were to decide which surface level features of the integration problems were most important, the network would be biased towards those elements.

Therefore I provided the network with the same features (operators and operands) that a human learner would see, and I gave equal representation to each one. Each integration problem is encoded with one input node per unary or binary operator, and a flag indicates whether the operator is actually present or not. This is known as binary value unit encoding, which is easier to learn than other encodings (Bishop 1995). At each location where a variable or a constant can legally occur, two units are used: one encodes whether there is a variable (e.g. x, y...) in that location, and the other whether there is a hard-coded constant (e.g. 1, 2, 3...). Integration problems are allowed to have either a constant or a lone variable, to which an operator could be applied, but not both. For example, 3 or x is legal, but $3x$ is not, because $3x$ it is an implied multiplication represented with an operator. Folding all variables into one node and all constants into another is a way of implementing prior knowledge. The values of the constants and variables do not affect the solution strategy. So neither the learner nor the network will spend time discriminating between them.

To create the input vectors for each integration problem, the input file containing the data set is passed through a script that converts each problem into postfix form. Each integration problem is restricted to a maximum of four terms. Each expression in a problem is represented by two terms: the element itself and any unary operator that is applied to it. For example: $\sin x$ is represented by the tuple x,sin. A simple number or variable in an expression is represented as the tuple x, *NONE*. The postfix terms are then passed through another script that maps the presence or absence of a given operator/operand to its location in the input vector. Short problems are padded with blanks. Thus the final result is a 55 unit vector, containing a series of 0s and 1s indicating which features comprise the calculus integration problem.

Summary of the data contained in the vector:

- Four 2-unit terms representing constants and variables
  Possible values: presence of a constant; presence of a variable such as x, y; no constant or variable present

- Four 8-unit Unary Operators
  Possible values:
  Trigonometric operators $\sin, \cos, \tan, \cot, \sec, \csc, \ln$, exponentiation $e(x)$

- Three 5-unit Binary Operators
  Possible values: multiplication, division, exponentiation ^, addition, subtraction

For example, the problem

$3 + cos(x) - sin(y) + ln(x)$

is coded as: 01 00000000 10 01000000 10 10000000 10 00000010 00010 00001 00010.

In more detail,

01 : No Variable; Constant (was 3)
00000000 : NONE (no unary operator for the constant)
10 : Variable (was the first x); No Constant
01000000 : cos (apply to the variable)
10 : Variable (was y); No Constant
10000000 : sin (apply to the variable)
10 : Variable (was the second x); No Constant
00000010 : ln (apply to the variable)
00010 : +
00001 : -
00010 : +

This coding scheme permits encoding hundreds (957) of Simple, Usub and Parts integration problems. The large number of problems should make the results generalizable. This coding scheme preserves the original surface level features. Therefore, the ANN is not biased by my choice of which features are important. The network sees the same visual features that a human learner sees and should be able to discover which features contribute most to conceptual understanding.

4 Experiments

This section explains the two hypotheses that were tested in the completed experiments. In section 4.1 the hypotheses are broken down into sub-points. Each sub-point describes a specific phenomenon that is part of the hypothesis, and allows detailed examination and precise testing of the hypothesis. Section 4.2 describes the experimental criteria that applied to both Hypotheses 1 and 2. This section provides the background information needed to understand descriptions of the individual experiments in Section 5.

4.1 Experimental Hypotheses

In the hypothesis descriptions below, the term "problem" is short for "calculus integration problem". A Problem Type is a category of solution strategy, i.e. Simple, Usub or Parts. "Long-term retention of material" is defined as the ability to remember a concept after other concepts have also been introduced.

Hypothesis 1: *An artificial neural network can be used as a model to investigate how people learn under different training scenarios.*

Hypothesis 1 validates the model; its sub-points are distilled from the literature on human learning. (Bransford et al. 2000,Chap. 3), (Duffy and Jonassen 1992,Chap 5), (Resnick and Ford 1981,Chap. 2). If the model exhibits these behaviors, we consider it a good model of human learning.

(1.1) During training, if problems that belong to one concept are introduced along with prob-

lems that belong to other concepts, error rates on tests are less than when concepts are introduced separately from one another.

(1.2) When concepts are reinforced inconsistently, only the most recently introduced concept is remembered.

(1.3) "Cramming", a brief intense review of all previously studied material done immediately prior to taking an exam, does not improve learning.

(1.4) Problem types that are related to one another are learned faster than unrelated problem types.

(1.5) Problem types that are related to one another become confused on tests, more than unrelated problem types.

Note that (1.5) is not inconsistent with (1.4). During learning, the network can draw upon prior exposure to features, and not start learning from scratch. Some solution categories will be eliminated very quickly. However, these same similarities between categories will cause some confusion on tests. How problems are related to one another will depend upon whether the learner is a novice or an expert. As cited earlier in this document, novice learners categorize based upon surface features, whereas experts categorize based upon underlying structural similarity. Therefore, the novice learners should confuse problems based upon clearly identifiable features. The expert learner should believe that a different set of problems are related to one another than the novice.

Hypothesis 2: *Different delivery methods result in different overall performance.*

After the first hypothesis is experimentally verified, Hypothesis 2 allows drawing practical conclusions from the model. This hypothesis focuses on how delivery methods help or hinder the learning of expert knowledge. Three delivery methods will be tested: two of them are inspired by existing methods of teaching students, and the third is inspired by machine learning research. (I will explain these methods in Sections 5, 6, and 7). This hypothesis is also broken down into sub-points.

(2.1) Delivery methods that use drill and test of separate problem types, will result in poor long-term retention of material.

(2.2) Delivery methods that force comparison of different problem types will result in longer-term retention of material than delivery methods that keep problem types separate from one another.

(2.3) Delivery methods that introduce new, increasingly complex, concepts along with reinforcement of old concepts, will result in the best long-term retention of material.

Note that (2.2) and (2.3) are related, but not the same. Sub-point (2.2) claims that any delivery method that forces comparisons would result in better retention than drill and test. Sub-point (2.3) suggests another such method, and claims that it would produce the best retention of all.

The experiments for testing Hypotheses 1 and 2 have been completed. In the next sections I will describe how these experiments were conducted.

4.2 Training and Test Criteria for all Experiment Types

This section describes the training and test criteria that applies to all of the completed experiments. There were three experimental scenarios, designed to test various aspects of Hypotheses 1 and 2. The first experiment type is Drill and Test (Section 5), the second is Fully Integrated (Section 6), and the third is Incremental Learning (Section 7).

As described in the previous section, the calculus integration problems were divided into paired training and test sets. For each experimental run, the training set was input to the network, using the 55 unit input vector, one problem at a time, in random order. The network was given each problem in a set once before any were repeated; one such round is called an "epoch". Each experimental run began with a new set of random weights between the nodes of the neural network. The interval for the random weight initialization was [-0.1, 0.1]. The learning rate for all experiments was .2 and the momentum was .9 . Each weight change was bounded to prevent it from becoming larger than the learning rate early in training. I used the Divergence error function, calculated by summing $t \log (t/o)$ over all the output nodes and the input examples, where t=target vector, o=output vector.

In each experiment, a network was trained on some combination of problem examples for a fixed amount of time. Time was measured in epochs. Because the focus of these experiments was upon performance, I experimented with different run times, searching for the best possible results in each experiment type. The best results for each scenario are reported, which means that the number of epochs run for each one makes no difference, as described in sections 5-7. In general, training lasted until training error either reached zero or plateaued somewhere before reaching zero.

Validation sets are often used in machine learning experiments to help identify when to cease training in order to achieve the best results. Validation does not apply to these experiments however. I was more interested in observing individual variations within experiment types, good or bad. Humans vary in their performance abilities, and in a formal educational environment such as a university, fast and slow learners are given the same amount of time to learn.

Each test item was an integration problem. The three output units in the network corresponded to the possible solution strategies: Simple Integration, Usubstitution, and Integration by Parts. There was always only one correct answer to a problem. This answer, called the "Best", was the answer suggested in a textbook, or by a calculus expert (faculty, TA, advanced student). For each test problem the network reported, via a percentage, how confident it was that the solution strategy was either Simple, Usub or Parts. If the confidence level for all solutions was below 80%, the problem was considered having "stumped" the network. The value 80% was chosen as a very rough approximation of "above average", much in the same way a standard grading scale of A-F often considers a B, in the 80+ range, to be attainable only by above average comprehension.

As I described earlier, the coded examples were divided into 10 or more experimental pairings of training-test data. Each of these pairings was executed at least ten times, randomly resetting the initial network weights each time. Therefore, each experimental scenario was executed at least 100 times. This repetition was done to look for natural variations in network learning of the same material, and to be able to test for statistically reliable differences. I performed t-tests upon the test scores, to look for statistically significant differences in performance between experiment types. In the following section I will describe in detail the first of the three experiment types, and the results that it produced.

5 Drill and Test Experiments

The experiments reported in this section tested Hypothesis 1.2-1.5 (see Section 4.1) and were inspired by a classic approach to presenting material to learners. Classic drill and test works as follows: methods of solving problems are introduced one at a time, for a fixed amount of time. Each successive problem type is more complex than the previous types. At the end of each method, the learner takes a midterm test. At the end of the time alloted for studying all concepts, there is a ”final exam” which includes all of the problem types. I expected to see good results on each midterm, and catastrophic forgetting on the final exam. I held this expectation because I attempted to model the poor retention exhibited by students who cram-memorize several related topics in succession, and later only retain well the most recent. So the hypothesis here was that the network/student would initially appear to learn each technique but subsequently be unable to retrieve all but the last one. The others would have been in short term memory and thus flushed out. The purpose of Hypothesis 1 was to validate the model and this goal was accomplished by the experiments.

5.1 Details of Training and Testing Regimen for Drill and Test Experiments

The network was trained first to identify Simple problems for 15 epochs, followed by Usub problems for 100 epochs and then Parts problems for 100 epochs. There was no overlap between problem types. Simple problems were trained for 15 epochs because this number always allowed the training error to reach 0%. After each training period, the network learner was tested with a single subject midterm. For example, after the end of training on Simple problems, the network took a test consisting of new Simple problems. After completing training on Parts problems, there was a Parts midterm followed by a comprehensive final exam. This final exam contained a mix of new Simple, Usub and Parts problems (SUP). All problems were designed as described in Section 3.

I also conducted 100 experiments with a slight modification to adjust the model to behave as if the learner undertakes a brief, short, comprehensive ”cram” session after the Parts midterm, and prior to the final exam. A cram session consisted of 30% of the previously studied problems, evenly distributed. A cram session lasted 22 epochs. Twenty-two epochs was 10% of the previous training time. I chose this time period to mimic a 2-day (10 hours total) crammed study session. Ten hours is approximately 10% of the 112 hours theoretically spent initially learning (16 week semester class that meets 5 days/week for one hour/day and requires 2 hours per week out-of-class study: 7x16=112 hours).

I conducted some experiments which shortened and lengthened the training segments. There was no effect on the results if training times were lengthened. There was also no effect on the results if training times were shortened, as long as those experiments that did not get stuck on a plateau were allowed to reach 0% (see next two sections). If they were not, performance was hurt. As with all the experimental scenarios, I report here the best results that were achieved.

5.2 Training Results for Drill and Test Experiments

Figure 2 shows the training error for the 100 experiments that included a cram session. In all Drill and Test experiments, the training error for the Simple problems showed an immediate rapid drop-off of error. The x-axis along the bottom of the figure shows what type of problem was being trained, and the x-axis along the top of the figure shows the epoch transition points (15, 115, 215). The error values along the y-axis are the Divergence Error (section 4.2). Because the Simple training part of

the curve is so small compared to the rest of the curve, I have also included a table below. Table 1 shows the average error during the 15 epochs of Simple training. The reader can see clearly in this table how rapidly the error values dropped and the network appeared to learn. This result seems logical: because Simple integration was the first concept seen by the network, and there were no competing problem types, it was easy for the network learner to acquire the desired output response.

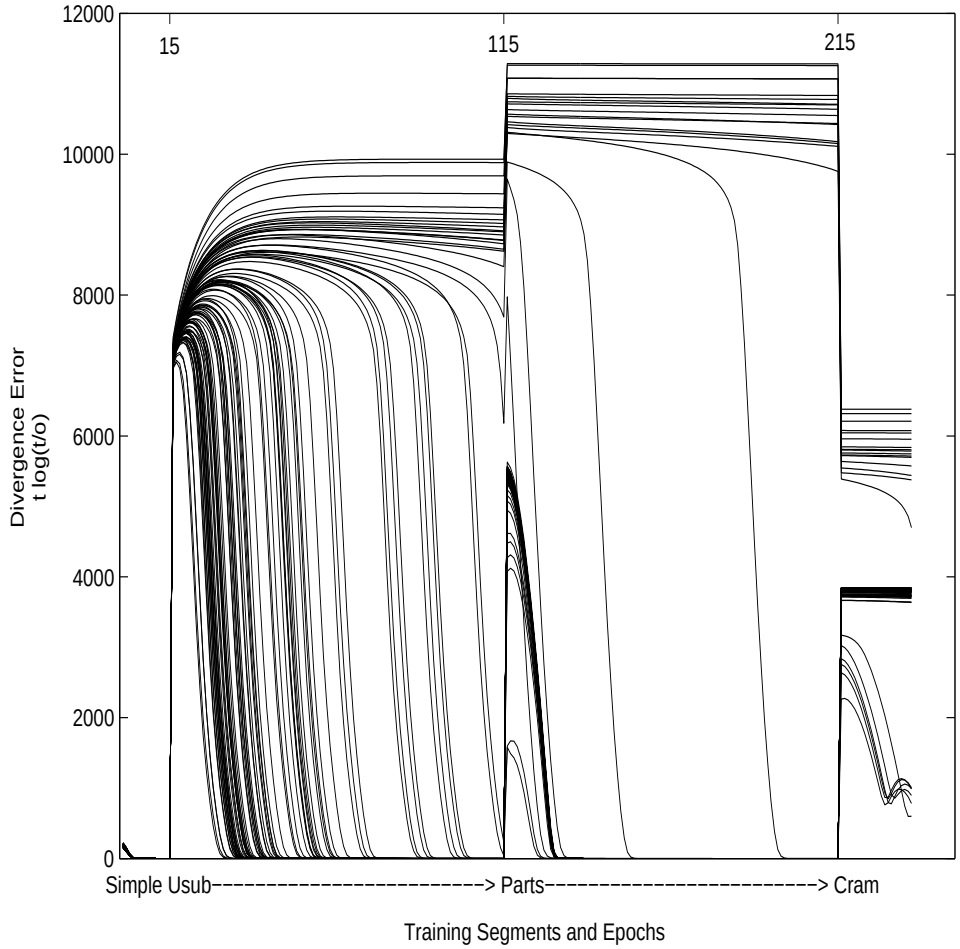

Figure 2: Drill and Test Learning - Training Error

As expected, the switch to Usub problems (epoch 15) produced an immediate jump to a very high error rate. The network learner had to ”shift gears” and adjust to a different concept. Even after the initial jump, the training error continued to climb for a short while. During this phase the network attempted unsuccessfully to fit new problems into its previously developed framework. Eventually, for most learners (83%), the network began to recognize and assimilate the new type of problem and there was a rapid drop in training error. Figure 2 shows how different learners took different amounts of time to follow this pattern of behavior. At the end of the Usub training some learners were still having high training error rates.

A similar bi-modal distribution of training error took place with Parts problems (epoch 115): an initial jump, followed for most learners with a sudden rapid reduction. However, those learners who had finished the Usub training with low errors, had lower and shorter periods of training error when Parts problems were introduced. On the other hand, those learners who had not fully grasped

Epoch	Average Training Error
1	202.6267
2	127.4555
3	54.6750
4	11.7953
5	1.3623
6	0.0967
7	0.00587
8	0.00041
9	0.000033
10	0.00000
11	0.00000
12	0.00000
13	0.00000
14	0.00000
15	0.00000

Table 1: Average Training Error During Simple Training

the Usub concept had higher and longer periods of training error when training switched to Parts problems.

The experiments without a cram session took the comprehensive final exam at the end of the Parts training (epoch 215). The experiments with a cram study session took the comprehensive exam after cramming. The cram session appeared to initially hurt those learners who were doing well on Parts problems alone: as can be seen, their training error rates shot up. On the other hand, the cram session appeared to help those learners who had been having the most trouble, as their training error rates dropped sharply. However, their training error rates did not drop as low as those of their counterparts who had had lower rates all along.

Looking at the training error for the whole experiment, most of the Drill and Test learners appeared able to rapidly and completely learn each new concept. Some Drill and Test learners however, appeared unable to learn the concepts at all. A few learners appeared able to learn, but slowly. It is interesting to note that the more trouble a learner had early in training, the more trouble they had later in training. These training results set up expectations of strong success on the midterm and comprehensive exams. In the next section, I report the results of these tests.

5.3 Testing Results for Drill and Test Experiments

Results for all Drill and Test experiments were virtually indistinguishable from one another. Simple midterms were always 100% successful, reflecting the reported 0% training error at the end of section training. The Usub and Parts midterms had a bi-modal distribution that paralleled the training error curves. The majority of learners, those whose error rates dropped to zero, were very successful on their midterms. Scores were often 100%, although occasionally the network made mistakes. Because these mistakes were rare, I can not draw any conclusions about them. Conversely, those learners who had high near-flat training error curves, scored 0% on their Usub and Parts midterms. These learners continued to select Simple as their answer to all test questions.

The opportunity to include a final cram session had no discernible effect on the comprehensive

exam (SUP) for either group of learners. The poorer learners scored on average 17.29% (standard deviation 4.95), continuing to always choose Simple as the answer. The high score for this group was 26.92%. The more successful group scored on average 41.65% (standard deviation 6.35), with a high score of 54.55%. With little exception, both sets of learners thought that all exam problems were Parts problems; their scores were the same as the percentage of Parts problems on their particular exams. They had lost the ability, apparently learned early on, to identify Simple and Usub. As predicted, the types of errors made on the exam reflected catastrophic forgetting of all but the most recently studied material. The difference in the average scores of the poor and successful learners was statistically significant, as evaluated with a t-test ($t = 17.5334, df = 27.985, p < 2.2e-16$).

All network learners exhibited high self-confidence, even when they were wrong. On the final exam, as on the midterms, the network did not have trouble choosing answers; it reported 100% confidence in its choices virtually all of the time.

5.4 Discussion of Drill and Test Experiments

The Drill and Test Experiments successfully tested a significant portion of Hypotheses 1 and 2. Hypothesis 2.1, Delivery methods that use drill and test of separate problem types, will result in poor long-term retention of material, were supported. Hypotheses 1.2 and 1.3, When concepts are reinforced inconsistently, only the most recently introduced concept is remembered, and that "Cramming" does not improve learning, were supported. Hypothesis 1.5, Problem types that are related to one another become confused on tests, more than unrelated problem types, was not supported in these experiments. The remainder of Hypothesis 2, and Hypothesis 1.1, are tested in Sections 6 and 7.

Hypothesis 1.4, Problems types that are related to one another are learned faster than unrelated problem types, was conditionally supported by the results. As was mentioned above, most network learners learned Parts problems far more easily than Usub problems. Because Parts problems are generally acknowledged to be more complex than Usub problems, I expected that they would be harder to recognize than Usub problems. Apparently, once the learner had figured out how to recognize Usub problems, it was easier to learn to recognize Parts problems. Those learners who had not managed to recognize Usub problems however, had even greater difficulty switching to Parts. Because the ability to recognize Usub problems affected the ability to recognize Parts problems, there is an implied relationship between Usub and Parts problems. What this relationship is, can not be determined without performing an analysis of the hidden layers. This analysis is part of my Proposed Work. That is why Hypothesis 1.4 can only be conditionally supported at this time.

Additional experiments are needed to find out more about this relationship. One question that needs to be asked is: do human learners perceive a similarity between Usub and Parts problems? It is very difficult to obtain this answer using traditional Drill and Test delivery methods. With humans, as in this model simulation, exams may not provide deep enough information. Subtle clues, in this case reports of error during training, need to be followed up with analysis of conceptual development. This analysis is part of my Proposed Work (Section 8.2).

6 Fully Integrated Experiments

These experiments tested Hypotheses 1.1 and 2.2 (see Section 4.1) and were inspired by the immersion experiences popular in foreign language instruction. In a full immersion situation, the learner is placed in an environment where they are completely surrounded by new, complex stimuli (for example: moving to a country where no one speaks English). Thus she or he has no choice but to sort out the most relevant language features in order to make effective communication decisions. Unfortunately, developing practical immersion experiments using humans is time consuming and costly. Very few, if any, controlled experiments have been done comparing human language immersion learning with classroom learning (for a detailed discussion and citations see (Spolsky 1989)). However, my computational model is ideal for an initial experiment with immersion learning. The cognitive mechanisms that enable a foreign language student to sort out important grammatical features should not be that different from those cognitive mechanisms that sort out features of mathematical grammar. Therefore, in these experiments I expected to see long term retention of concepts that was far better than in the Drill and Test experiments.

6.1 Details of Training and Testing Regimen for Fully Integrated Experiments

The network was trained on all of the problem types (Simple, Usub, Parts) simultaneously. This means that a mix of Simple, Usub and Parts problems were input to the network in random order. At the end of each epoch (one pass through all the problems), the problems were shown again in a new random order. Training took place for 215 epochs; this is the same number of epochs as were run in the Drill and Test experiments without cramming. By 215 epochs, the training error had reached a plateau.

All of the exams were SUP (i.e. cumulative) exams, to reflect the mixed nature of the training. Initially, I gave midterms at the same epochs as in the Drill and Test experiments, but initial results showed such a marked difference in behavior (compared to Drill and Test) that I decided to give midterms after every epoch in order to more closely monitor progress of the network learner.

I also ran some experiments that extended the training time beyond 215 epochs. Neither training error nor test scores were affected.

6.2 Training Results for Fully Integrated Experiments

Training behavior was highly consistent for all Fully Integrated experiments (see Figure 3). The training type is labeled across the lower x-axis. In order to be able to compare these results with Drill and Test more easily, I have also labeled epochs 15, 115, and 215 along top and bottom x-axes. Also as in Figure 2, the y-axis contains values for the Divergence training error. As can be seen in Figure 3, the training error began with a jump in error, and after a brief fluctuation, dropped dramatically and smoothly. Training error reached a plateau, well above 0, although there continued to be minor error fluctuations. In contrast to the Drill and Test training error, Fully Integrated training error was highly unimodal, and low.

6.3 Testing Results for Fully Integrated Experiments

The graph of test scores for Fully Integrated experiments looks very much like an inverted graph of the training error (see Figure 4). In order to show the results more clearly, this figure displays

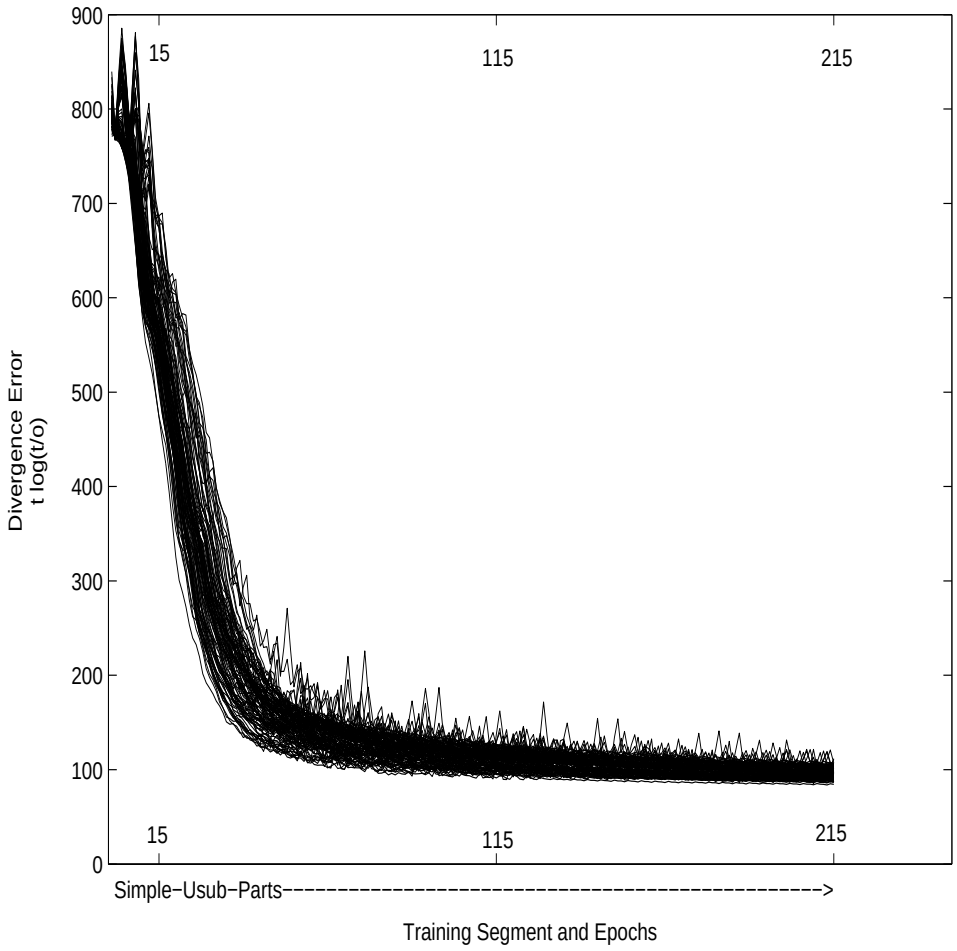

Figure 3: Fully Integrated Learning - Training Error

a fully representative subset of the 100 experimental runs. Test scores are on the y-axis, and the test type (SUP) is shown along the x-axis, along with the epochs seen on previous figures. As can be seen, the learners initially failed the midterms miserably, but then their scores rapidly increased, and finally plateaued. Scores fluctuated as they rose, but eventually smoothed out. The average final score was 76.99%, and the standard deviation was 7.94 . The best score was 80.76% .

The types of errors made on tests, and confidence levels for all answers, have not been fully analyzed yet. This data has been collected, and I propose to complete the analysis as part of my dissertation (see Section 8 - Proposed Work)

6.4 Discussion of Fully Integrated Experiments

The Fully Integrated experiments successfully tested Hypotheses 1.1 and 2.2. Hypothesis 1.1, During training, if problems that belong to one concept are introduced along with problems that belong to other concepts, error rates on tests are less than when concepts are introduced separately from one another, was supported. Hypothesis 2.2, Delivery methods that force comparison of different problem types will result in longer-term retention of material than delivery methods that keep problem types separate from one another, was supported.

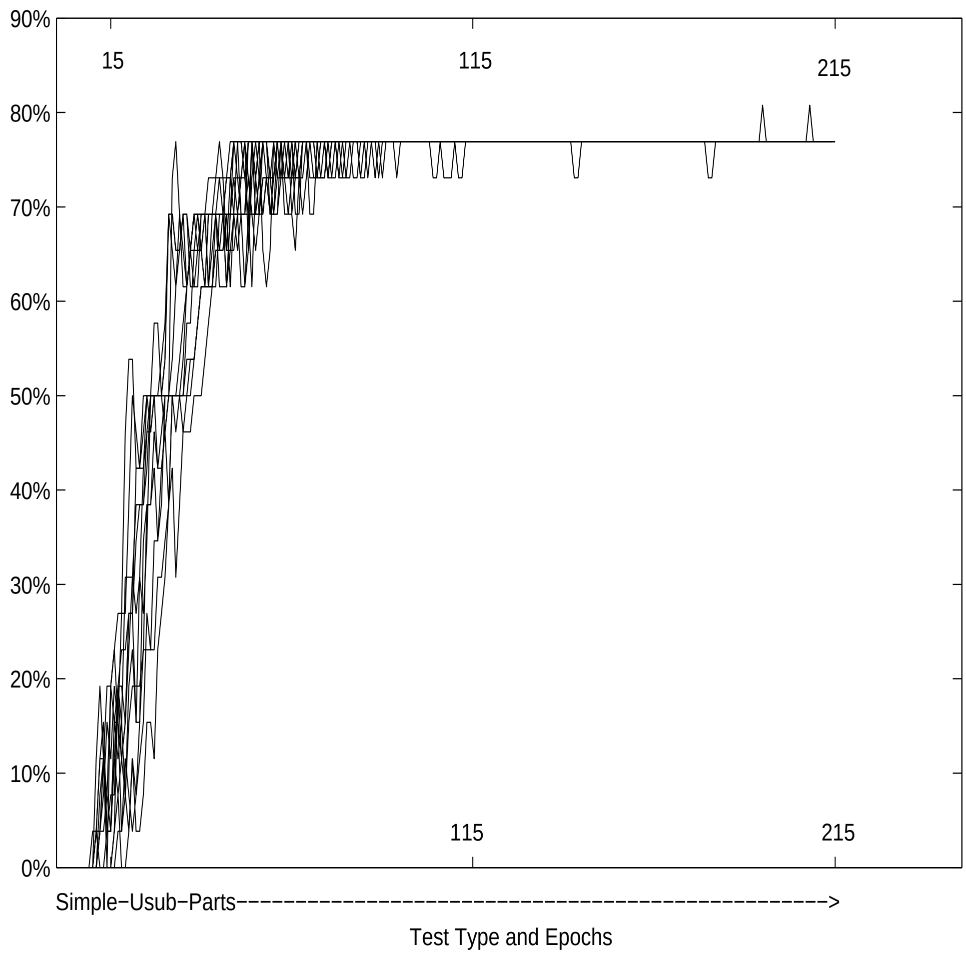

Figure 4: Fully Integrated Learning - Test Scores

These results are very different from the Drill and Test results. Hypotheses 1.1 and 2.2 support the idea that complex training improves learning, whereas making the learning task too easy may be counterproductive. Although the final test scores for Fully Integrated experiments never reached 100%, it may be unreasonable to ever expect them to do that. How realistic is it to assume that a learner would consistently get perfect scores on any exam of truly challenging intellectual material? Also, as we saw in the analysis of Drill and Test results, midterm scores of 100% were misleading. They implied a level of interim learning and understanding which was not supported when the final exam required the learner to distinguish complex concepts.

An analysis of test errors will provide more information about what the learner understands on each test. It will be interesting to see if Hypothesis 1.5 is supported by the Fully Integrated experiments (problem types that are related to one another become confused on tests). A preliminary look at error and confidence data indicates that the types of learner errors are going to be more complex than in the Drill and Test experiments. This is to be expected because a Fully Integrated delivery forces the learner to compare all the problem types at once.

As with the Drill and Test experiments, I propose to analyze the conceptual development in the hidden layers (see Section 8.2). Already, prior to having those results, this immersion-inspired approach to learning is promising. The results from these experiments have provided enough in-

formation to begin a conversation about how one might train math students in this manner. I will discuss this idea further in Section 8.4.

7 Incremental Learning Experiments

These experiments were designed to test Hypotheses 1.1, 2.2 and 2.3. They were inspired by an understanding in the machine learning community that it is often most effective to tackle large computational tasks by starting with small problems and gradually increasing their complexity (Elman 1991b). Often, highly complex computational problems cannot be resolved with a direct assault. The difficulties are not limitations in physical resources, but the large number of variables - their unknown dependencies and interactions. The problem becomes a statistical one; when there are a large number of co-dependent variables, it is extremely hard to discover the role that each one plays in the problem and its solution. Given that the acquisition of intellectual expertise is very complex and poorly understood, an incremental learning approach should produce better quality learning and understanding than a direct approach. Thus, in these experiments I expected to see long term retention that was better than in either the Drill and Test or Fully Integrated experiments.

7.1 Details of Training and Testing Regimen for Incremental Learning Experiments

The experiments reported in this section ran for a total of 140 epochs. As with the Drill and Test experiments, there were three training periods. Each training period gradually increased the complexity of the learner's categorization problem. The network was first trained for 5 epochs to identify Simple problems. During the second training period, Usub problems were added to the Simple problems. Training on the Simple-Usub mix ran lasted an additional 30 epochs. For the third training period, Parts problems were added. Training on the Simple-Usub-Parts mix lasted an additional 105 epochs.

As with the Fully Integrated experiments, I initially gave midterms only at the training period change points and at the very end of training (epochs 5, 35, 140). Later, I added midterms after every epoch in order to look more closely at performance changes. During the first training period, only Simple midterms were given; during the second training period, mixed Simple-Usub (SU) midterms were given, and during the third training period, all midterms were mixed Simple-Usub-Parts.

The training times used in these experiments are different from those used in the other experiments. Preliminary results running for 215 epochs (as in the other experimental scenarios) were highly successful, and unlike Drill and Test experiments, performance was affected by altering the transition points (recall that Fully Integrated experiments have no transition points). In the next section I will discuss how altering the transition points affected performance, and in the Discussion sections (Section 7.4 and Section 9.3) I will speculate on why these particular transition points worked the best.

7.2 Training Results for Incremental Learning Experiments

As expected, the learning behavior in the Simple-only training segment was the same as in the Drill and Test experiments: the training error curve showed an immediate drop-off (Figures 5 and 2).

When Usub problems were added for the second period of training (epoch 5), the training errors jumped. This jump was not nearly as high as when Drill and Test switched completely to Usub, but

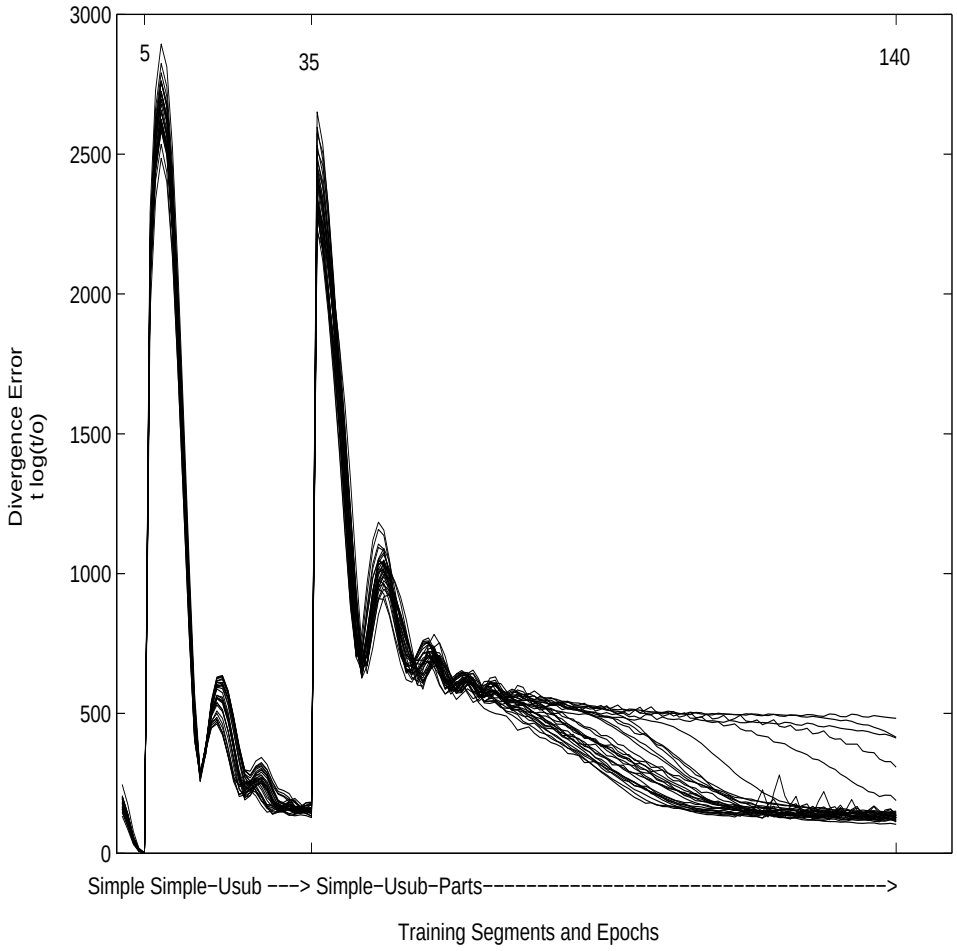

Figure 5: Incremental Learning - Training Error.

lower than when Fully Integrated Learning began. Training error then rapidly dropped, although it never reached zero. When Parts problems were added for the third segment of training (epoch 35), error again leaped and dropped, showing the same pattern: an initial rapid drop followed by a continuing, fluctuating descent. As in the second segment of training, errors plateaued above zero. In contrast to the Drill and Test training error, and more like the Fully Integrated training error, Incremental Learning training error was very nearly unimodal. As can be seen in Figure 5, almost all of the 100 experiments plateaued close together. Final training errors for Fully Integrated learning were very close to those of Fully Integrated learning.

In the experiments where I made the training segments longer, little changed. Regardless of how long the network trained for the SU and SUP segments, training errors never reached zero. However, test results were mildly but negatively effected by substantially longer S or SU training (as discussed in the next section). If the SU training was shortener, the SUP segment took substantially longer to drop to its plateau. If the SU training segment was substantially shortened, SUP training errors fluctuated more and took much longer to decrease.

7.3 Testing Results for Incremental Learning Experiments

Figure 6 shows a fully representative subset of the test scores for the Incremental Learning experiments. The types of midterms and the epoch transition points (5, 35, 140) are shown along the x-axis. The test scores are on the y-axis. As can be seen on the figure, Simple-only midterms very rapidly reached scores of 100%. When Usub problems were introduced (epoch 5), test scores began to fluctuate severely. Scores would drop to, or near, zero, rebound, and then drop again. Over time, although the fluctuation continued, the overall scores increased. In a few cases, midterm scores reached 100%, however the majority of cases peaked at 70-75%.

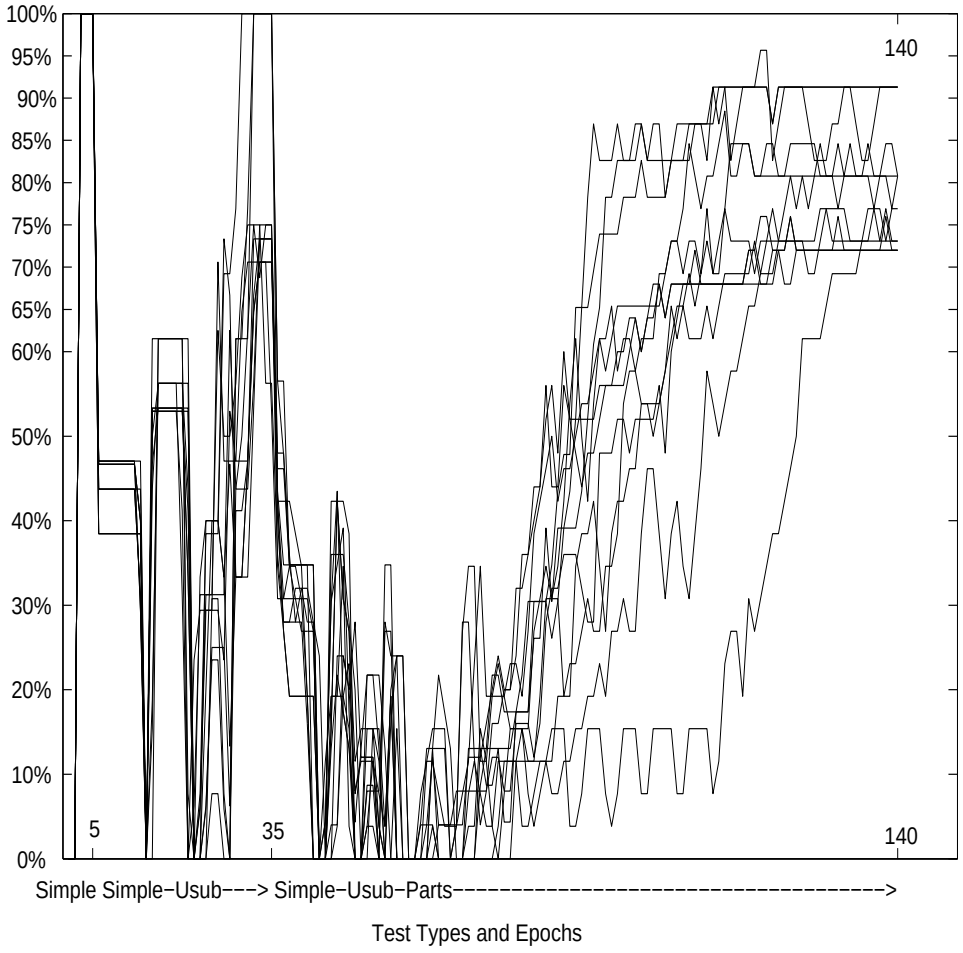

Figure 6: Incremental Learning - Test Scores

When Parts problems were introduced for the third training period (epoch 35), the pattern of fluctuating scores was accentuated. Midterm scores immediately plummeted, although it is interesting to note that even the downward drop was often not smooth, but marked by brief plateaus and recoveries. Performance continued to deteriorate for longer than in the SU training segment, with scores fluctuating lower and lower. In contrast to the SU midterm scores, SUP midterm scores appeared to tighten in closer and closer to complete failure (for a while nearly all midterms fluctuated well under 20%). Eventually, performance began to improve, with prominent individual differences. Eventually, virtually all midterm scores surpassed the 70% point, averaging 81.9%, with a standard deviation of 8.23.The maximum midterm score was 95.6%, higher than any score

reached in a Fully Integrated learning experiment. As evaluated with a t-test, the Incremental Learning final exam scores were higher than those of the Fully Integrated learning ($t = 1.9574, df = 11.869, p - value = 0.07423$).

The types of errors made on tests, and confidence levels for all answers, have not yet been fully analyzed. The data has been collected, and I propose to complete this analysis as part of my dissertation (Section 8.3).

As I briefly mentioned in the previous section, longer periods of training did not always translate to increased performance. In fact, there was a balance between doing enough training, and doing too much training. Although extended training on the Simple and Simple-Usub segments (up to and beyond the "lowest error") did not show overt signs of over-training, midterm test scores were higher when training did not run well beyond these points. This evidence of over and under training might have been revealed with the use of a validation test set, however, as I explained earlier (Section 4.2), the influence of a validation set would have interfered with my desire to model human-like individual variations in learning. Finally, as with Drill and Test and Fully Integrated experiments, I experimented with extending the final training segment to see if scores would continue to rise: they did not.

7.4 Discussion of Incremental Learning Experiments

The Incremental Learning experiments successfully tested Hypotheses 1.1, 2.2, and 2.3. Hypothesis 1.1, During training, if problems that belong to one concept are introduced along with problems that belong to other concepts, error rates on tests are less than when concepts are introduced separately from one another, was supported. Hypothesis 2.2, Delivery methods that force comparison of different problem types will result in longer-term retention of material than delivery methods that keep problem types separate from one another, was supported. Hypothesis 2.3, Delivery methods that introduce new, increasingly complex, concepts along with reinforcement of old concepts, will result in the best long-term retention of material, was supported.

As I pointed out in the previous section, the results for Incremental Learning were very different from either Drill and Test or Fully Integrated learning. Incremental Learning appears to benefit from having both structure and high complexity. By imposing structure upon the order in which problem types are introduced during training, the network learner is allowed to focus on the characteristics of a smaller number of problem types at the beginning of learning. However, the learner is not allowed to become "complacent" and overconfident early on. Just as the first concept (Simple problems) is acquired, additional problems (Usub) are mixed in. The resulting confusion is apparent in the fluctuating training error and midterm scores. Over time, as the learner grapples with the two contrasting problem types, confusion (as measured by fluctuation) diminishes and midterm scores rise. When Parts problems are introduced, it becomes vastly more difficult to discriminate between the concepts. This added complexity causes yet another drop in confidence and midterm scores. As would also be expected in a classroom situation, it is far more difficult to compare three related problem types than two. So the confusion lasts longer and is more difficult to resolve. Also, individual learner differences can be expected to become more apparent with more challenging concepts. Fortunately, the "priming" effect of the previous training segments allows most Incremental Learning learners to eventually do well. In most cases they do better than the Fully Integrated learners. Scores never reach 100%, yet, as I explained in Section 6.4, it is unrealistic to assume that highly challenging and interrelated intellectual concepts, could ever be "perfectly" understood. An expert recognizes the deep multidimensional nature of problems in her or his domain. This perspective

will sometimes lead to incorrect choices. So this less than perfect result validates the behavior of the model.

8 Proposed Work

With the completion of the Incremental Learning Experiments, all parts of Hypotheses 1 and 2 have been tested. Hypothesis 1 validated the model: An ANN can be used as a model to investigate how people learn under different training scenarios. Hypothesis 1 was fully supported in three of its sub-points (1.1, 1.2, 1.3) and conditionally supported in another (1.4). One sub-point (1.5) was not supported by the Drill and Test experiments, but still needs to be examined for the Fully Integrated and Incremental Learning experiments. Hypothesis 2, Different delivery methods result in very different overall performance, was fully supported. In this section I will describe how I propose to continue my work on this foundation. Sections 8.1 and 8.2 describe the work I intend to complete first. Section 8.3 discusses a number of possible further directions: which ones I pursue depends on earlier results. Section 8.4 discusses further directions that this dissertation may initiate.

8.1 Analysis of Test Errors

The first task is to complete the analysis of test errors for the Fully Integrated Learning and Incremental Learning experiments. This analysis is a necessary foundation for the other proposed work directions. The data has already been collected. I propose to do the same analysis as I have already completed for Drill and Test experiments. This analysis includes conducting a quantitative analysis of what types of errors (and their frequency) were made on critical midterms. In addition, I plan to simultaneously analyze the confidence with which correct and incorrect answers were chosen. These results will complement the test scores, and provide greater insight into the learners' performance on exams. They will also further test Hypothesis 1.5, Problem types that are related to one another become confused on tests, more than unrelated problem types.

8.2 Test Hypothesis 3

The bulk of my Proposed Work consists of testing Hypothesis 3. Hypothesis 3 will analyze the conceptual development that took place in each of the experimental scenarios. This information will provide information about learning that adds to what has been learned from the training error and test scores. Training and test data measure information that the Instructor (or researcher) requested. Conceptual development directly reflects the learner's internal perspective. We need to analyze this perspective in order to fully understand the effect of each delivery strategy on the learner.

8.2.1 Details of Hypothesis 3

Hypothesis 3: *Different delivery methods will result in different internal conceptual representations and conceptual development.*

Hypothesis 3 has three sub-points:

(3.1) Delivery methods that emphasize Drill and Test will produce internal network representations that differ greatly depending upon what problem type has been recently studied.

(3.2) Delivery methods that introduce concepts using Incremental Learning, will produce internal network representations that change gradually over time.

(3.3) Delivery methods that introduce concepts using Incremental Learning, will result in better conceptual development than either Drill and Test or Fully Integrated learning.

In order to analyze the conceptual development that took place during the Drill and Test, Fully Integrated, and Incremental Learning experiments, the hidden units of the neural network will be examined. As I explained in Section 3.1, the hidden nodes of a neural network contain data that represent a concept or concepts. These data change over time as the network learns. Thus it is important to analyze what concepts the network has at various stages of learning.

For the Drill and Test experiments I propose to begin by analyzing the hidden layer at critical stages: the beginning of training, at transition points (S to U, U to P), at the beginning of the cram session, and at the end of training. This will allow me to align stages of conceptual development with critical midterm exams. I will also analyze the hidden layer at epochs that appear to be critical based upon the training errors. For the Drill and Test learners, these include epochs when a plateau begins (for those having trouble learning). This will allow me to compare conceptual development between those learners who rapidly acquired the concept and those who did not. I also plan to compare the conceptual development during the cram session for the two groups of learners.

For the Incremental Learning experiments I propose to begin by analyzing similar critical points: beginning of training, transition points (S to SU, SU to SUP) and the end of training. This will allow me to not only monitor development of these learners, but also to make comparisons with the Drill and Test learners. As with Drill and Test, I will also analyze the hidden layer at epochs that appear critical during learning. For Incremental Learning these points include the peaks and valleys of fluctuations during SU and SUP training segments. In addition I may discover other potential critical points after I have completed the analysis of test errors.

For the Fully Integrated Learning experiments, there are critical points at the beginning and ending of training. There is also an apparent critical point when the training error begins to plateau. These points are the ones I propose to analyze first. I also plan to sample the conceptual development at one or two points along the plateau and along the initial steep descent. I will choose the exact epochs after I complete the analysis of test errors. This analysis will hopefully provide information about where interesting conceptual changes are occurring.

8.2.2 Using ICA to Analyze the Hidden Layer

I propose to conduct my analysis of the hidden layer using Independent Component Analysis (ICA). ICA is a relatively new clustering algorithm - it was developed within the last 10 years. It is neurally based, and has shown promise in some applications where Principal Component Analysis (PCA) has failed. Most of the applications of ICA to date have been in engineering (multiple signal discrimination for example) although there have been a few attempts to use it for linguistic analysis and data mining (Bingham, Kuusisto, and Lagus 2002),(Isbell and Viola 1999),(Kolenda, Hansen, and Sigurdsson 2000). We have installed a version of ICA, called FastICA (Hyvärinen and Oja 1997). I

will provide a brief description of the ICA algorithm in the next section.

In parallel with the ICA analysis, I will use PCA. PCA has been widely used for analyzing the hidden layers of artificial neural networks. So PCA results will be easy to interpret and use in comparing with prior work. It will also allow me to contrast the capabilities of the ICA tool with a method of known properties.

8.2.3 ICA Method

Because it is so new, I am including this technical section describing briefly how ICA works. This section assumes a basic understanding of statistical terminology and concepts. It also assumes some understanding of matrix manipulations, and some familiarity with theories of neural algorithms. For a fuller explanation see (Hyvärinen, Karhunen, and Oja 2001).

ICA Preprocessing Step 1: The input vectors are first centered (normalized), to have zero mean and unit variance (z-scores). Then a Covariance matrix is calculated from this normalized input matrix. A Covariance matrix is an exploratory tool, showing all data variation in relation to each other.

Step 2: Perform an Eigen Analysis (aka Eigen-value Decomposition- EVD) and PCA upon the Covariance matrix.

EVD orthogonalizes (i.e. decorrelates) the components of the Covariance matrix. As a result, two more $N * N$ matrices are produced: an Eigen-value matrix, and an Eigen-vector matrix.

PCA orders the components so that those with the largest components come first. PCA also provides the option to eliminate components with least variation, which reduces noise as a result.

Performing (Fast)ICA After Preprocessing FastICA (Hyvärinen and Oja 1997) is the name of the implementation of ICA that I propose to use in my analysis. There are several competing implementations of ICA, and each one has a slightly different algorithm. I chose FastICA because it converges faster than other ICA algorithms, and does not require the user to pick a good learning rate. These properties mean that in most situations, FastICA will produce results more quickly and reliably than its competitors. The following discussion then, best describes FastICA.

ICA assumes that even after decorrelation and accounting for variance, an additional transformation of the data is necessary to reveal truly independent components. This is particularly true in the case of data with a non-Gaussian distribution. When data has Gaussian distributions, independence and uncorrelatedness are equivalent; for non-Gaussian distributions, they differ, and independence is the stronger property.

Thus, preprocessing is not theoretically necessary, but it reduces the computational load for ICA. By starting to work with pre-processed data, the ICA algorithm can more rapidly converge on a solution that other methods are unable to find.

A fundamental assumption of ICA is that the independent components are statistically independent (i.e. $P(AB) = P(A) * P(B)$). Another assumption of ICA is that the independent components are non-Gaussian (per above note). In practice, if only one of the independent components is Gaussian, ICA will still work. If several components are Gaussian, then ICA will be unable to identify

them. If all of the components are Gaussian, then ICA will perform as well as PCA but no better. Another assumption of ICA is that the number of independent components is equal to the observed mixtures. If this assumption doesn't hold, it is usually because the data is highly multidimensional. In this event, PCA can be used during preprocessing to reduce the dimensionality of the data.

There are two fundamental steps of all ICA methods. The first step is to formulate a valid contrast function. Solutions are found at the minima or maxima of the function. Statistical estimation of the minima or maxima are based upon optimization algorithms. Many (but not all) optimization algorithms are based upon gradient learning algorithms. The second step is to use an algorithm for estimating the free variables of the system. There are two principal theoretical approaches. One is based on Estimation Theory, the other on Information Theory. FastICA is a neural fixed-point algorithm (not gradient based). It used an iterative weight-update scheme in order to obtain convergence.

The bottom line is that ICA can take very complicated multi-dimensional data, and identify its underlying structure. It can solve problems that are too complicated for other methods of statistical estimation. For example, ICA was originally developed to tackle the "cocktail party problem". Imagine that you have several audio-tapes that were made at a party full of people. You would like to separate out one or more of the original voices. This is very difficult, because you do not know the positions of the microphones in relation to the people. However, ICA has had a lot of success with applications of this type, which are known as Blind Signal Separation problems. Analyzing the complex data in the hidden layers of an ANN is a good opportunity to test ICA further.

8.3 Possible Directions

There are some additional experiments that I would like to do depending on how the results of the ICA analysis turn out. The following subsections introduce these ideas.

8.3.1 Different Encoding

Although the encoding scheme I have developed was constructed in order to minimize external bias, it would be useful to create a different encoding scheme for the same data. Then, results for the same delivery types could be compared with a different coding scheme. The primary reason for making this comparison would be to find out what effect my original coding scheme had on the results presented in this document. This new coding scheme would also have to be cognitively justifiable. As with the current scheme, it would have to reflect what a human sees, which is not necessarily what would be mathematically the most efficient or elegant.

8.3.2 Increase Test Set Sizes

As I explained in previous sections, the number of problems in each test set was determined as a proportion of the number of problems in the training set that it was paired with. The proportion of training set examples to test set examples was similar to that likely to be seen by a human in a formal learning situation. This has meant that test sets were sometimes quite small. If I loosened this restriction and created larger test sets (relative to training set size) it might be possible to more easily see patterns of test behavior. However, this proposed work might prove unnecessary once I have completed analysis of the current test errors.

8.3.3 Create a Harder Conceptual Problem

One of the higher goals of successful calculus teaching is to enable students to make connections between integration and differentiation. For each integration strategy there is a corresponding differentiation strategy. Historically, math students have been taught a course in integration and a course in differentiation. In many schools, the courses can be taken in either order. The connection between the two classes is left for the student to figure out. If, after having completed both classes, a student can look at an expression and identify underlying mathematical principles linking integration and differentiation, she or he will be able to more easily acquire advanced mathematical concepts that rely upon subtleties of both.

It would be interesting to extend my work to include this type of harder conceptual problem. The idea is to first train the network from two different perspectives: identification of integration solution strategies, and identification of differentiation strategies. I propose to see if the network can acquire the ability to look at a problem in one format (say, integration) and identify the paired solution strategy (differentiation).

This task would require the creation of a set of differentiation problems and their corresponding solution strategies. A coding scheme similar to the one for integration problems would have to be developed. Then the input layer of the ANN would have to be changed to hold either type of problem. Ideally, the nodes in the input layer would be able to do double duty, holding either an integration problem or a differentiation problem. The coding scheme for differentiation problems will determine if this is possible. If not, the input layer will instead consist of one set of nodes for integration problems and another set of nodes for differentiation problems.

There would be six output nodes in the new network, because the output layer would include both differentiation and integration solution strategies. For each test problem, the correct response would be to identify two answers equally, the integration solution strategy and the differentiation solution strategy that corresponds to it. This is a fundamentally different test procedure, as the desired goal on a test question would be to spread the output activation across two nodes equally, rather than choose one, as in the current experiments.

For the architecture I intend to use a recurrent network. Recurrent networks have been successful in modeling input processes that have dependencies over time (Cottrell and Tsung 1993). My network will remember matched integration-differentiation pairs. The network will be trained by inputting first one of the pair, and then the second. The recurrent network will be able to save the input from the first (say, integration) problem, and then link that input with the second (differentiation) problem that comes in the next time step. If all of the data is trained in pairs, the network should learn to identify underlying regularities between the pairs.

8.4 Future Work

The work described in this section is work that I would like to continue later on. This work takes the results reported in this document into new areas, working directly with humans, and applying the results to other domains of expertise. All of the future work has been made possible by the completed experiments.

8.4.1 Studies on Humans

There are several approaches to extending these results to studies with adults. Perhaps the best way to begin would be with lab experiments. As I discussed early in this document (Section 2.2.1) it is extremely difficult to conduct controlled studies of realistic adult learning. However, now that interesting learning behavior has been identified with three well defined delivery approaches using this model, these same approaches can be investigated with people. One approach would be to enlist three groups of volunteers, matched as well as possible in terms of prior exposure, motivation and ability. These students could then be given calculus integration problems, with their solution categories, to analyze, in whatever way they desire. Their analysis would be restricted only by available time and the order in which they see the types of problems.

Another human study that needs to be done is to find out if learners perceive a similarity between Usub and Parts problems. Results from the Drill and Test experiments implied that there is an underlying relationship between these two types of problems (Section 5.4). Students who are currently taking calculus classes could be interviewed to elicit a holistic view of their conscious and sub-conscious understanding. It would be important to interview learners who are apparently unsuccessful as well as successful.

8.4.2 Other Domains of Expertise

As I discussed in the Introduction and Background sections of this document, there are many features of expertise that cross domains. Therefore, it will be important to apply this model to domains other than calculus. There are many potential directions; the one that appeals to me the most is Computer Science. This is a logical choice as well because Computer Science is a mathematically based discipline (Beaubouef 2002). In addition, there is a large body of literature about student learning problems in Computer Science (for a few examples see (Chalk 2001),(Fleury 2000),(Haberman and Averbuch 2002),(Ginat 2003)). Thus, as with mathematics, there would be much information with which to validate the model.

9 Perspectives from Learning Theory

This section is the first of two extended Discussion sections (the other is Section 10). The purpose of this section is to discuss similarities between my results and psychological theories of human learning. These theories are well known to educational researchers, but researchers in other fields are often not aware of them. Because a primary motivation for my research is to improve student learning, it is useful to see how my results support established theories of human learning, and how these theories predict my results. In Section 9.1 I discuss the parallels between the Drill and Test results and Behaviorism. In Section 9.2 I discuss how theories of second language learning can shed light on the results from the Fully Integrated Experiments. In Section 9.3 I speculate on the relationship between the Incremental Learning results and Constructivism.

9.1 Drill and Test Experiments

Apparently, all Drill and Test learners were only able to retain one concept in their memory. For the majority of learners, this was the concept seen most recently. Previous concepts were not integrated into long-term memory, but were flushed out. As each new problem type was introduced, the

network received feedback (visible on Figure 2 as increased training error) that existing decision-making strategies were no longer successful. Thus, there was no immediate incentive to retain them. From a psychological perspective, there is more incentive to forget early material. Once a subject is no longer needed, it symbolically places ”closure” upon that material. With humans, midterms reinforce this closure.

As can also be the case with human learners, single-subject midterms say nothing about long term conceptual understanding, i.e. retention, generalization and transfer of previously learned material. In fact, as shown in these studies, they may provide misleading or falsely encouraging data. A learner may believe, and thus report, that they have learned the material, when in fact they have merely learned what the expected answer is at any given time. It is noteworthy that in these experiments the network learners made their incorrect choices very confidently. This false confidence implies that the network learner was unaware that it had a problem, and was responding mechanically to each question. Only when we examine the results of the comprehensive final exam, with or without last minute cramming, do we discover the magnitude of the learning problem.

This false confidence and forgetting resembles some important aspects of Behaviorist psychological learning theory. Classic studies in Operant Conditioning showed that behavior increases in frequency if it is followed by a positive reward (Skinner 1938/1999), (Thorndike 1898). When that reward is removed, the behavior rapidly drops to extinction. It is possible to conceptualize what is happening in Drill and Test from this perspective. Each problem set is reinforced for a period of time, only to be displaced by a new one. As only one form of input is introduced at a time, the output choice becomes almost mechanical: ”when presented with a problem, choose Simple (Usub, Parts)”. There is no motivation to learn at any other level. When the network learner takes the SUP exam, it continues to respond to each problem with the answer that was most recently rewarded (Parts). Because it is no longer receiving feedback, the network repeats the same mistake over and over again.

Perhaps the most interesting question arising from the results of the Drill and Test Experiments concerns the fundamental nature of ”learning”. Within the educational research and psychological literature, definition of this term has been and continues to be controversial; the results from these studies also bring up questions about what learning is. For example, just what has been ”learned” at the time of the midterms? If the learner performs well on a midterm yet later forgets the material, was the material really ever ”learned”? What does it mean if the learner is unaware of being confused? What concepts is the learner creating when she or he encounters material delivered with the Drill and Test technique? With humans, these and other questions can be investigated via the use of individualized assessments, interviews or other qualitative investigation. Equivalently, with a computational model, we can perform a hidden layer analysis. Such an analysis is part of my Proposed Work (see Section 8.2). Meanwhile, the evidence presented so far justifies the use of the term ”apparent learning”. From this point forward, I will use this term to refer to situations in which there is evidence of short term or surface level recognition of problems, but the learner is not fully integrating important concepts.

Many years ago, psychologists viewed mathematics learning as a collection of isolated skills (Thorndike and Skinner 1922), but more recently, studies show that instructing with this approach results in poor conceptual understanding (Carpenter, Corbitt, Kopner, and Lindquist 1980), (Resnick and Ford 1981). Therefore, even though some of the Drill and Test results may appear ”obvious” from a machine learning perspective, the results are worth thinking about. That similar behavior has been reported in human studies not only supports the cognitive validity of the model, it begs the question as to why so much instruction continues to be delivered using this technique?

9.2 Fully Integrated Experiments

Fully Integrated learners are practicing and absorbing information in an unguided environment. By its very nature, it is difficult to study unstructured learning. However, there is a great deal of literature and theory about second and foreign language learning which can provide an interesting perspective on these results. Although "natural language learning" (as unguided full immersion is referred to in the second language learning literature) can be very effective, an extensive number of studies show that learning improves even more if some structure is imposed in the early stages of learning (for a review see (Spolsky 1989)).

The results from my Fully Integrated experiments appear to fit well with the literature on second language learning cited above. After some initial difficulty (seen as early training error), the Fully Integrated learners perform well when immersed in all the integration problems. Also, learning was even better when full immersion was preceded by early structure (as in the Incremental Learning Experiments - see next section). Like with natural language learners, the network is probably confusing similar-appearing answers; we will get a fuller picture of what these errors are like after I complete the analysis of errors.

9.3 Incremental Learning Experiments

Incremental Learning learners followed more independent paths during training than either Drill and Test or Fully Integrated learners. This individual variation became more prominent as the concepts became more difficult to learn. Some learned more quickly than others. All of them alternated better test scores with worse test scores, as they appeared to struggle with new material. Over the long run, all of these learners were able to achieve acceptable or excellent scores on their comprehensive exams.

Psychological learning theory suggests that these results reflect human learning behavior. Piaget's well known theory of learning via equilibration is achieved by a process of assimilation and accommodation (Gruber and Voneche 1995). These two actions do not take place in a predictable order. The mind fluctuates back and forth between them, as new experiences provide an influx of external perturbations. During this process, the learner feels confused because she or he is experiencing cognitive contradictions. These contradictions, and the confusion, continue until eventually the mental concepts stabilize ("equilibrate")(Gruber and Voneche 1995). Although most of Piaget's life and work predated modern advances in AI and ANNs, others have presented strong evidence that Piaget himself would have been supportive of computational approaches to psychological modeling (Boden 1989).

Equilibration is at the heart of, and is a precursor of, modern Constructivist theories of learning. The Constructivist perspective views learning as a self-regulatory process of struggle and conflict. Existing cognitive concepts have to negotiate with discrepant new insights. The process of negotiation eventually produces new meaning (Fosnot 1996). This cognitive behavior resonates with the results seen in my studies, and the behavior of connectionist models in general. As I described earlier (Section 3.1), ANNs gradually learn by making guesses, receiving feedback about how wrong they were, adjusting internally, and trying again. Large amounts of input data eventually allow the network to make very accurate guesses, however, in the beginning the network will making a lot of large mistakes. The test scores in the Incremental Learning experiments also fluctuate widely up and down in the beginning, appear to be doing worse and worse overall, but then begin a fairly steady improvement (Figure 6).

Many educational theorists with a Constructivist orientation also believe in the existence of a "Zone of Proximal Development"(ZOPED) (Vygotsky 1978). The ZOPED is a psychological point on the very edge of a person's current understanding of some concept. If new material is presented below the ZOPED, deep analysis is not required and surface (or no) learning may result. If new material is beyond the ZOPED, the learner is overwhelmed or confused. The implication is that an ideal way to teach is to scaffold the delivery of material at the learner's ZOPED. This will force the learner to struggle with new material in a structured supportive environment. At first the learners will have problems, but eventually they will grasp the new concept. Then the ZOPED moves upwards and the process can be repeated.

The Incremental Learning experiments completed here are an example of scaffolded delivery using a ZOPED. Concepts start out easy (Simple integration), and gradually become more complex (Usub, then Parts). The learner is forced to compare old concepts with new concepts, but only one new concept is added at at time. The moment at which a new concept is introduced has been planned carefully, so that it happens just as the learner has begun to fully understand the current concept(s) (as measured by the point when training error reached zero or a low plateau). Each transition point can be thought of as occurring in a ZOPED. As I reported earlier, if the transition happened earlier, the learner took longer to learn the next concept. In this situation, the learner was overwhelmed by new information it was not ready for. If the transition occurred later, performance on tests was worse. In this situation, the learner had over-trained; in human terms, the learner had over-analyzed the problems, and read too much into the examples.

There are other areas as well, in which studies of learning and growth have reported that complex learning may sometimes result in temporary regressive behavior. One of these areas is infant development. For example, we know that as vision develops, the eye tries to identify increasingly complex details of what it sees. An apparent information overload occurs, and when this happens, infants revert temporarily to simpler behaviors (Cohen 1998). Another example involves motor development. Research has shown that when infants begin to walk, cognitive overload occurs and interferes with arm usage. Thus they temporarily revert from one armed to two armed reaching until walking is integrated into their skill set (Corbetta and Bojczyk 2002). Finally, models of species evolution have shown that highly complex yet general behavior evolves more successfully if tasks are learned incrementally (Gomez and Miikkulainen 1997). These results from studies in widely different domains support the results of my experiments showing that that Incremental Learning produces the best learning of complex tasks.

10 Perspectives from Symbolic Cognitive Science

This is the second of two extended Discussion sections (the other is Section 9). The purpose of this section is to contrast my approach with the symbolic cognitive models used previously to study expertise. Symbolic and connectionist models are the two computational approaches used to model higher-order thinking. This document has focused on the connectionist approach used for my experiments. It is useful to also discuss the symbolic approach in order to understand my results in the context of cognitive studies as a whole. In Section 10.1 I provide a historical context for symbolic cognitive science. In section 10.2 I introduce the primary types of symbolic models of cognition, and in section 10.3 provide examples of symbolic models that have been applied to the study of expertise and mathematics learning.

10.1 The Historical Context of the Symbolic Paradigm

Symbolic computational models have long been used successfully to study higher-order thinking tasks. In order to understand why they have such a long history, and why my approach of using an ANN is unique, one must look at the history of artificial intelligence (AI).

The belief that cognition is a form of computation, is neither new, nor unique to computer science. This perspective predates 20th Century studies in artificial intelligence by hundreds of years. Symbolism, the paradigm which has dominated cognitive science and AI, is socially and historically grounded in the writings of the philosopher Thomas Hobbes (17th C.). Hobbes stated unequivocally: ”by RATIOCINATION I mean computation”, which he explained as meaning that thinking consists of symbol operations that function most clearly and rationally when following methodical rules ((Hume 1739/1977) and as cited in (Haugeland 1985)) .

Hobbes’ ideas were echoed and expanded upon by subsequent philosophers including David Hume, who introduced an explicitly mechanical theoretical perspective. The influence of Hume is clearly visible in the development of 20th Century cognitive psychology. Jerome Bruner, a pioneer in the field, reflected the dual influences of philosophical thought and the emerging field of computer science in the introduction to ”A Study of Thinking”: ”our own work has progressed from initial concern with the number of bits of information assimilated...to an ultimate concern with the informational properties of long sequences of acts called strategies...” (Bruner, Goodnow, and Austin 1956). These writings reflect the continued assumption that human cognition is symbolic and operates via externally driven and sequential mechanisms.

Early computer science pioneers of cognition such as Herbert Simon were strongly influenced by Bruner in their early writings. Like Bruner, Simon believed that there are few intrinsic characteristics of thinking - most of what occurs is due not to biological properties but to environmental input and subsequent adaptation (Simon 1969/1981). Simon explicitly placed the human mind and brain together with the computer as members of the family of physical symbol systems. Furthermore, he described memory and language as list structures operated upon by a serial processor (Simon 1969/1981).

This approach to the study of cognition dominated the AI community for several decades following the Second World War. Although alternative theories have since emerged, many AI researchers still prefer symbol-based theories of cognition. Given that the symbolic paradigm has been around for so long, it is no surprise that computational models have also been predominantly symbolic. Symbolic models have been very successful in modeling many attributes of expertise. However, some model actual human behavior better than others. The difficulty that symbolic computational models have mimicking human behavior, is that they are generally not robust when there are no known rules and algorithms; they are unable to self-reflect and reorganize internal processing based upon new information; they cannot identify unanticipated goal states or patterns. These limitations make it difficult to model the human process of acquiring intellectual expertise.

In order to demonstrate this difficulty and opportunity, the next two sections will describe some of the dominant types of symbolic models and how they have been applied to studying expertise and mathematics learning.

10.2 Types of Symbolic Models of Cognition

Stimulus-Response (S-R) models represent one of the earliest approaches to modeling cognition. They are based upon classical psychological behaviorism and have been used with some success in

the development of robots that learn to navigate unknown environments. These systems excel when all possible inputs and outputs are known, and when there are direct relationships between them. (A very interesting and enlightening look at early work in this area is described by Murray (1955)). However, strict S-R models are inherently unable to generalize problem-solving behavior and thus cannot mimic higher-order thinking such as intellectual expertise. For this reason, most simple S-R models have been augmented with rule-based systems.

Rule-based systems expand upon simple S-R models in that they utilize vast data stores and anticipated scenarios to guide decision-making. These systems, also known as expert systems, have been extensively deployed in industrial environments. They typically use extensive knowledge bases to provide input to a decision making process. This process contains information about which specific sequences of actions will lead to desired solutions. Examples typical of these systems can be found in medicine (Warren, Warren, and Freedman 1993), law (Woodin 2001), engineering (Reed 2000), and reading comprehension (Dyer 1983). However, expert systems depend upon logical rules; they do not perform well when desired outcomes are based upon statistical correlations. In addition, although they are domain experts, expert systems focus on achieving a goal state; they are less concerned with the learning process. Thus they do not model learning as it occurs in humans. These qualities prevent expert systems from being used to model the cognitive processes utilized by a human expert.

Various numerical methods have been used to address reasoning and learning under uncertainty. Often they have drawn upon probability theory, especially the Bayes rule. These inference networks have been utilized to study domains where prior probabilities are available and consistent, such as process control (Tsoukalas, Lee, and Ragheb 1989). Another numerical method is Fuzzy Logic, which is based upon theories of cluster analysis. Fuzzy logic has been successful in control systems such as wireless networks (Shen, Mark, and Ye 2000). Numerical uncertainty methods are generally good at prediction and control, and analyzing properties of data sets. Successful such applications include speech recognition (Bergadano and Giordana 1986) and hardware level controllers (Salapura and Hamann 1996). However, numerical uncertainty approaches cannot make good decisions without consistent prior probabilities. As with expert systems, these models focus on reaching a goal state and are not concerned with the human learning process. They are poorly suited for analyzing and formulating hypotheses about intuitive problem solving, an important ability that emerges in human experts.

10.3 Symbolic Models of Expertise and Mathematics Learning

Several symbolic models for expertise have been developed. Some of these models focus on mathematics learning; others mimic human memory and recognition more generally. The purpose of this discussion is to illustrate, through example, that symbolic models reflect a different school of thought about human cognition than connectionist models do. The models reviewed in this section provide examples of the philosophical contrast between the symbolic and connectionist paradigms. The models in this section usually prioritize efficiency, or achieving a goal state, over understanding the human learning process.

The LEX system built decision trees (DT) to classify and solve calculus integration problems (Mitchell 1983), (Mitchell and Ranan 1983). LEX learned by creating practice problems, presenting hypothesized solutions, solving the problem, and then working backwards from the solution to form a heuristic. This system has much in common with the operations I am using with an ANN. Mitchell refers to Integration by Parts, Simple Integration and Usubstitution as operators; my model refers

to them as solution strategies. Both LEX and my system learn what ”operator” is appropriate for a given integration problem. Both models seek a low-cost solution and neither is guaranteed to find it because of the classic problem of local minima. Both evaluate their performance during processing: the DT by introducing negative examples, and the ANN by adjusting weights during training. A partially learned heuristic in LEX roughly corresponds to the hidden layers of the ANN during training.

On the other hand, LEX has drawbacks as a model of human behavior. LEX creates its own training examples to help it learn. This is highly efficient, but not representative of the majority of human learners, whether they are in the classroom or studying alone. Also, LEX learns by solving the problems and working backwards to form a heuristic. My interviews with mathematics faculty and TAs indicate that this is a common pedagogical tactic. Unfortunately, this approach to training ignores evidence, from those same interviewees and previously cited literature, that experts do not usually learn strategies by working backwards.

ACT-R is another symbolic model of learning which has been applied to mathematics. ACT-R is an ongoing project, originating in the early 1980s. The mission of the research group is broad (”to develop a cognitive architecture that can perform in detail a full range of cognitive tasks”) but their goals differ from mine. Many of the ACT-R applications have been intelligent tutors for mathematics education and computer programming. A tutor reacts to an outside learner; it does not itself attempt to learn mathematical concepts. In contrast, I am attempting to model what is happening inside the learner’s mind.

A somewhat different symbolic approach to modeling expertise was taken by Mooney (1993). He posited that ”people acquire expertise through a process of abstract instruction and experience with specific cases” (Mooney 1993). Feature information is critical for concept learning in Mooney’s model. Feature data belongs to one of two categories: ”explanatory” or ”non-explanatory”. Explanatory data is data that does not aid in proper categorization. Such data may be irrelevant or obscure; in any case it is not covered by any known rule for application (Mooney 1993). These statements are certainly appealing. Feature data is important for categorization as is learning via examples. Mooney’s implementation of these ideas however, assumes, as does Mitchell, that there are rules for all problem-solving situations; if there is no known rule, either the rule must be found or the data is not helpful. An alternate perspective, the one I take in my model, is that data that may appear irrelevant is often engaged in subtle interactions with other data - this is how complex intellectual schema are formed.

In general, the symbolic models just described focus on reaching a desired goal state in the most computationally efficient manner possible. Their focus is not on the human learning process. In part this choice was made by previous researchers. It is easier to study something that is composed of facts and behaviors, than it is to study an unknown process. Symbolic models are an important step in the study of expertise; we cannot reach a goal state (in this case expertise) if we do not first know what it looks like. Now that we have some of that knowledge, it is possible to place focus on the process of learning it. My model contributes to both computer science and cognitive science by doing so.

11 Conclusion

In this proposal I have shown results supporting the following two hypotheses: 1) An artificial neural network can be used as a model to investigate how people learn under different training scenarios

2) Different delivery methods result in different overall performance. These results provide new insight into how humans learn complex cognitive tasks. In my Proposed work I intend to test a third hypothesis: 3) Different delivery methods will result in different internal conceptual representations and conceptual development.

The results reported in this study, and the proposed dissertation work, will contribute to many fields. For computer science, and artificial intelligence in particular, this research expands the body of existing computational models of human learning and provides a new example of modeling higher-order cognition. We have discovered that priming an ANN can produce higher performance than inputting all the types of data randomly. This discovery can lead to re-evaluating the training design of existing computational models in which performance and efficiency are the primary goals.

The results reported in this study have also given cognitive science new understanding of how conceptual development takes place in different types of learning. Educators now have additional evidence that structured, integrated delivery methods are better for learners than oversimplification and isolation of learning tasks. In addition, this work encourages educators and cognitive scientists to focus on analyzing problem examples, to see what their underlying structure contributes to learner categorization and subsequent problem solving.

References

Abudiab, M. (2001). The impact of technology on teaching an ordinary differential equations course. *JCSC*, 16(3):7–18.

Alvarez, P., and Squire, L. (1994). Memory consolidation and the medial temporal lobe: A simple neural network model. In *Proceedings of the National Academy of Sciences*, vol. 91, 7041–7045.

Atkin, J. (1998). The OECD study of innovations in science, mathematics and technology education. *Journal of Curriculum Studies*, 30(6):647–660.

Beaubouef, T. (2002). Why computer science students need math. *Inroads, The SIGCSE Bulletin*, 34/4:57–59.

Bednar, J. A., and Miikkulainen, R. (2003). Learning innate face preferences. *Neural Computation*, 15(7). In press.

Bergadano, F., and Giordana, A. (1986). A framework for knowledge representation and use in pattern analysis. In *Proceedings of the ACM SIGART international symposium on Methodologies for intelligent systems*, 423–431.

Bingham, E., Kuusisto, J., and Lagus, K. (2002). ICA and SOM in text document analysis. In *SIGIR*.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Boden, M. (1989). *Artificial Intelligence in Psychology*. MIT Press.

Bransford, J., Brown, A., and Cocking, R. (2000). *How People Learn: Brain, Mind, Experience and School*. N.R. Council National Academy Press.

Bruer, J. (1997). Education and the brain: A bridge too far. *Educational Researcher*, 26(8):4–16.

Bruner, J., Goodnow, J., and Austin, G. (1956). *A Study of Thinking*. John Wiley and Sons, Inc.

Bruner, J., and Kenney, H. (1965). Representation and mathematics learning. In Morrisett, and Vinsonhaler, editors, *Monographs of the Society for Research in Child Development serial 99*, vol. 30-1. University of Chicago Press.

Burton, L. (1999). Mathematics and their epistemologies – and the learning of mathematics. In *1st International Conference of European Research on Mathematics Education*.

Carpenter, T., Corbitt, M., Kopner, K., and Lindquist, M. (1980). Results of the second NAEP mathematics assessment: Secondary school. *Mathematics Teacher*, 73(5):329–338.

Chalk, P. (2001). Scaffolding learning in virtual environments. In *Proceedings of the 6th annual conference on innovation and technology in computer science education*, 85–88. ACM Press.

Chaput, H. H., and Cohen, L. B. (2001). A model of infant causal perception and its development. In *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*, 182–187. Erlbaum.

Chase, W., and Simon, H. (1973). Perception in chess. *Cognitive Psychology*, 4:55–81.

Chen, C.-C., and Miikkulainen, R. (2001). Creating melodies with evolving recurrent neural networks. In *Proceedings of the 2001 International Joint Conference on Neural Networks (IJCNN-2001, Washington DC)*, 2241–2246. IEEE.

Chi, M., Feltovich, P., and Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5:121–152.

Cohen, L. B. (1998). An information–processing approach to infant perception and cognition. In *The Development of Sensory, Motor, and Cognitive Capacities in Early Infancy*, 277–300. East Sussex:Psychology Press.

Corbetta, D., and Bojczyk, K. E. (2002). Infants return to two-handed reaching when they are learning to walk. *Journal of Motor Behavior*, 34(1):83–95.

Cottrell, G., and Tsung, F. (1993). Learning simple arithmetic procedures. *Connection Science*, 5(1):37–58.

Cromer, A. (1997). *Connected Knowledge: Science, Philosophy and Education.* Oxford University Press.

DeGroot, A. (1966). Perception and memory versus thought: Some ideas and recent findings. In Kleinmuntz, B., editor, *Problem Solving: Research, methods and theory*, 19–50. John Wiley.

Dreyfus, H., and Dreyfus, S. (1986). *Mind Over Machine: The Power of Human Intuition and Expertise*. The Free Press, Macmillan, Inc.

Duffy, T., and Jonassen, D. (1992). *Constructivism and the Technology of Instruction: A Conversation.* Lawrence Erlbaum Associates.

Duffy, T., Lowyck, J., and Jonassen, D. (1993). *Designing Environments for Constructive Learning.*, vol. 105 of *N.S.A. Division NATO ASI Series F: Computer and Systems Sciences*. Springer-Verlag.

Dyer, M. (1983). *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*. MIT Press.

Editor, A. (2000). American heritage dictionary of the english language.

Eisenhart, M., Finkel, E., and Marion, S. (1996). Creating the conditions for scientific literacy: A re-examination. *American Educational Research Journal*, 33(2):261–295.

Elman, J. (1991a). Distributed representations, simple recurrent networks and grammatical structure. *Machine Learning*, 7:195–225.

Elman, J. (1991b). Incremental learning, or the importance of starting small. *Cognitive Science*, 443–448.

Ferrini-Mundy, J., and Graham, K. (1994). Research in calculus learning: Understanding of limits, derivitives, and integrals. In *Research Issues in Undergraduate Mathematics Learning*, MAA Notes Number 33, 31–45. MAA.

Fleury, A. E. (2000). Programming in java: student-constructed rules. In *Proceedings of the thirty-first SIGCSE technical symposium on computer science education*, 197–201. ACM Press.

Fosnot, C. (1996). Constructivism: A psychological theory of learning. In Fosnot, C., editor, *Constructivism: Theory Perspectives and Practice*, 8–33. Teachers College Press.

Garner, R. (1990). When children and adults do not use learning strategies: Toward a theory of settings. *Review of Educational Research*, 60(4):517–529.

Ginat, D. (2003). The greedy trap and learning from mistakes. In *Proceedings of the 34th technical symposium on computer science education*, 11–15. ACM Press.

Glaser, R. (1987). Thoughts on expertise. In Schooler, C., and Schaie, W., editors, *Cognitive Functioning and Social Structure Over the Life Course*, 81–94. Ablex.

Gomez, F., and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342.

Gruber, H. E., and Voneche, J. J., editors (1995). *The Essential Piaget*. Aronson.

Haberman, B., and Averbuch, H. (2002). The case of base cases: why are they so difficult to recognize? student difficulties with recursion. In *Proceedings of the 7th annual conference on innovation and technology in computer science education*, 84–88. ACM Press.

Haugeland, J. (1985). *Artificial Intelligence: The Very Idea*. MIT Press.

Hayes, J. (1989). *The Complete Problem Solver*. Lawrence Erlbaum Associates.

Haykin, S. (1999). *Neural Networks, A Comprehensive Foundation*. Prentice Hall.

Hinsley, D., Hayes, J., and Simon, H. (1977). From words to equations: Meaning and representation in algebra word problems. In Just, M., and Carpenter, P., editors, *Cognitive Processes in Comprehension*, 89–108. Lawrence Erlbaum Assoc.

Hume, D. (1739/1977). *A Treatise of Human Nature*. J.M. Dent and Sons LTD.

Hyvärinen, A., Karhunen, J., and Oja, E. (2001). *Independent Component Analysis*. John Wiley and Sons.

Hyvärinen, A., and Oja, E. (1997). A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9/7:1483–1492.

Isbell, C. L. J., and Viola, P. (1999). Restructuring sparse high dimensional data for effective retrieval. In Kearns, M. S., A., S. S., and Cohn, D. A., editors, *Advances in Neural Information Processing*, vol. 11, 480–486. Morgan Kaufmann.

Kolenda, T., Hansen, L. K., and Sigurdsson, S. (2000). Independent components in text. In *Advances in Independent Component Analysis*, 235–256. Springer.

Lang, S. (1986). *A First Course in Calculus*. Springer-Verlag. Fifth edition.

Marshall, R., and Tucker, M. (1992). *Thinking for a Living - Education and the Wealth of Nations.* Basic Books: A Division of Harper Collins.

McClelland, J. L., McNaughton, B. L., and 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(3):419–457.

Medley, M. (2000). Why do they fail? How do they succeed? The success of some students in CS2. In *2nd Annual CCSC on Computing in Small Colleges Western Conference*, 218–224. JCSC.

Miikkulainen, R. (1997). Natural language processing with subsymbolic neural networks. In Browne, A., editor, *Neural Network Perspectives on Cognition and Adaptive Robotics,*, 120–139. Institute of Physics Publishing.

Mitchell, T. (1983). Learning and problem solving. In *IJCAI*, 1139–1151.

Mitchell, T., and Ranan, B. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine Learning*, 163–190. Tioga Publishing Co.

Mooney, R. (1993). Integrating theory and data in category learning. In Nakamura, G., Medin, D., and Taraban, R., editors, *Categorization by Humans and Machines*, vol. 29, 189–218. Academic Press.

Moore, T., and Wick, M. (1994). Assessing student's critical thinking skills and attitudes toward computer science. In *The twenty-fifth technical symposium on computer science education*. ACM.

Murray, F. (1955). Mechanisms and robots. *Journal of the Asssociation of Computing Machinery*, 2:61–82.

Norman, F., and Prichard, M. (1994). Cognitive obstacles to the learning of calculus: A krutetskiian perspective. *Research Issues in Undergraduate Mathematics Learning*, 65–77.

Owen, E., and Sweller, J. (1989). Should problem solving be used as a learning device in mathematics? *JRME*, 20(3):322–328.

Paris, S., and Winograd, P. (1990). How metacognition can promote academic learning and instruction. In *Dimensions of thinking and cognitive instruction*. Lawrence Erlbaum.

Reed, D. (2000). A perceptual assistant to do sound equalization. In *5th international conference on Intelligent user interfaces*, 212–218. ACM.

Resnick, L., and Ford, W. (1981). *The Psychology of Mathematics for Instruction.* Lawrence Erlbaum Associates.

Robinson, C., and Hayes, J. (1978). Making inferences about relevance in understanding problems. In Revlin, R., and Mayer, R., editors, *Human Reasoning*, 195–206. V.H. Winston and Sons.

Rohde, D. (v2.3.1). lens, the light, efficient network simulator.

Rosenblatt, F. (1958). The perceptron: A probalistic model for information storage and organization in the brain. *Psychological Review*, 65(6).

Ross, B., and Spalding, T. (1991). Some influences of instance comparisons on concept formation. In Fisher, D., Pazzani, M., and Langley, P., editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, 207–236. Morgan Kauffman.

Rumelhart, D. (1998). The architecture of mind: A connectionist approach. In Thagard, P., editor, *Mind Readings*, 207–238. MIT Press.

Rumelhart, D., and McLelland, J. (1987). Learning the past tenses of english verbs: Implicit rules or parallel distributed processing. In MacWhinney, B., editor, *Mechanisms of Language Acquisition*. Lawrence Erlbaum Associates.

Salapura, V., and Hamann, V. (1996). Implementing fuzzy control systems using VHDL and statecharts. In *Proceedings of the conference with EURO-VHDL '96 and exhibition on European Design Automation*, 53–58.

Schoenfeld, A., and Herrmann, D. (1982). Problem perception and knowledge strucuture in expert and novice mathematical problem solvers. *Journal of Experimental Psychology: Learning, Memory, Cognition*, 8:484–494.

Selden, A., and Seldon, J. (1997). Should mathematicians and mathematics educators be listening to cognitive psycholgists?. online.

Selden, J., Selden, A., and Mason, A. (1994). Even good calculus students can't solve nonroutine problems. *Research Issues in Undergraduate Mathematics Learning*, 19–26.

Shen, X., Mark, J. W., and Ye, J. (2000). User mobility profile prediction: an adaptive fuzzy inference approach. *Wireless Networks*, 6(5):363–374.

Silver, E. (1979). Student perceptions of relatedness among mathematical verbal problems. *JRME*, 10(3):195–210.

Silverman, R. A. (1985). *Calculus with Analytic Geometry*. Prentice-Hall.

Simon, H. (1969/1981). *The Sciences of the Artificial*. The MIT Press. Second edition.

Skinner, B. (1938/1999). *The Behavior of Organisms*. B.F. Skinner Foundation.

Spolsky, B. (1989). *Conditions for Second Language Learning: Introduction to a General Theory.* Oxford University Press.

Stewart, J. (1995). *Calculus*. Brooks Cole Publishing. Third edition.

Stigler, J., and Hiebert, J. (1999). *The Teaching Gap*. The Free Press.

Thorndike, E. (1898). Animal intelligence: An experimental study of the associative processes in animals. *Psychological Review: Monograph Supplements*.

Thorndike, E., and Skinner, B. (1922). *The Psychology of Arithmetic*. MacMillan.

Tsoukalas, L., Lee, G., and Ragheb, M. (1989). Anticipatory monitoring and control in a process environment. In *The second international conference on industrial and engineering applications of artifical intelligence and expert systems*, vol. 1, 278–287.

Valverde, G., Bianchi, L., Wolfe, R., Schmidt, W.H., and Hwang, R. (2002). *According to the Book: Using TIMSS to Investigate the Translation of Policy into Practice through the World of Textbooks.* Kluwer Academic Publishers.

Viscuso, S., Anderson, J., and Spoehr, K. (1989). Representing simple arithmetic in neural networks. In Tiberghien, G., editor, *Advances in Cognitive Science*, vol. 2, 141–164. Ellis Horwood and John Wiley and Sons.

Vygotsky, L. (1978). *Mind in Society.* Harvard University Press.

Warren, J., Warren, D., and Freedman, R. (1993). A knowledge-based patient data acquisition system for primary care medicine. In *Second international conference on information and knowledge management*, 547–553.

Woodin, D. (2001). Design and implementation of gungaweb: an application of a classical expert system technology to the production of web-based commercial systems. In *The 8th international conference on artificial intelligence and law*, 104–108.