here is that the Language Acquisition Device may only be able to hypothesize very simple rules. The rules can interact to produce lengthy derivations, and they are extensively chunked during development to arrive at adult linguistic performance. But chunking is the only source of complex rules; they cannot be created *de novo* by the LAD.

Why have rules at all in a connectionist theory? Rules separate *policy* (what Chomsky calls linguistic "switch settings") from *mechanism* (the fundamental ability to do insertions, deletions, and mutations.) If a mechanism such as the String Editing Network is universal and genetically determined, then the LAD's job is tremendously easier: it can concentrate on learning just the policies of the speaker's language.

This paper makes no assumption that policies require explicit symbolic representations in speakers' heads. Rather, it shows that chunking can occur even when there is no working memory trace available and new rules cannot be constructed symbolically. The connectionist chunker acquires its rules incrementally, through self-supervised backpropagation and rehearsal of prior knowledge. Further experiments are planned to analyze the representations the chunker develops.

Acknowledgments

This work was supported by a contract with Hughes Research Laboratories, by National Science Foundation Grant EET-8716324, and by the Office of Naval Research under contract number N00014-86-K-0678. I thank Deirdre Wheeler, Marco Zagha, Michael Witbrock, and George Lakoff for discussions that contributed to this work, and Gillette Elvgren for help with the simulations.

References

Newell, A. (1987) The 1987 William James Lectures: Unified Theories of Cognition. Given at Harvard University.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987) Soar: An architecture for general intelligence. *Artificial Intelligence* 33(1):1-64.

Pinker, S., and Prince, A. (1988) On language and connectionism: analysis of a parallel distributed processing model of language acquisition. In S. Pinker & J. Mehler (eds.), *Connections and Symbols*. MIT Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland and (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press.

Rumelhart, D. E., and McClelland, J. L. (1986) On learning the past tense of English verbs. In J. L. McClelland and D. E. Rumelhart (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 2. MIT Press.

Smolensky, P. (1988) On the hypotheses underlying connectionism. *Behavioral and Brain Sciences* **11**(1).

Touretzky, D. S. (1989) Towards a connectionist phonology: the "many maps" approach to sequence manipulation. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum Associates.

A PDP model of sequence learning that exhibits the power law

Yoshiro Miyata
Bell Communications Research

ABSTRACT

This paper examines some characteristics of the learning process in a model of skill learning (Miyata, 1987) in which performance of executing sequential actions becomes increasingly more efficient as a skill is practiced. The model is a hierarchy of sequential PDP networks which was designed to model a shift from a slow, serial performance of a novice to a fast, parallel performance of an expert in tasks such as typing. The network develops representation of a set of sequences as it tries to produce the sequences faster. The model was found to yield the power law of learning (Newell and Rosenbloom, 1981). In addition, it exhibited a frequency effect on substitution errors similar to what was found in typing (Grudin, 1983).

INTRODUCTION

Learning has intrinsic importance to the study of skilled performance because the nature of performance dramatically changes as a skill develops. However, study of skill learning is difficult because one has to explain not only what processing structure underlies skilled performance in a particular task domain, but also what mechanism enables us to build such structures as a result of experience in many different tasks. The approach taken in this work is to look for phenomena that are observed across a wide range of tasks and to try to develop a model of action learning that attempts to account for what seem to be quite general phenomena.

I have previously proposed a model of skill learning in which performance of sequential actions becomes faster as a skill is practiced (Miyata, 1987). This model successfully accounted for some effects of presentation frequency in typing, specifically the effect on speed (Grudin & Larochelle, 1982) and on a class of execution errors (Sellen, 1986). This paper reports on some additional experiments which revealed some interesting characteristics of the learning process in the model. In particular, the model is shown to exhibit the power law of learning. In addition, it exhibited a frequency effect on error patterns similar to typing errors at the keystroke level as well as at the sequence level as previously shown (Miyata, 1987). I will start by describing an example of the power law to illustrate the kind of skills being modeled in this work.

The Power Law

Probably the most general phenomenon we know about learning is that practice makes performance faster. However, more specific regularities seem to exist. For a wide variety of tasks, the learning curve (i.e., a plot of the time to perform the task versus the number of trials) produces approximately a straight line in log-log coordinates (Newell & Rosenbloom, 1981). This has been generally called the power law

because it indicates that the time is a power function of the number of trials. Figure 1 shows a learning curve of a beginning typist in a typing class studied by Gentner (1983) with the median interstroke interval plotted against the number of weeks of study on a log-log scale.

THE HS MODEL

Consider a person learning to type: first, a novice who has just started to learn typing. Suppose the learner has an intention to type the word "type". He would first find the key "t" on the keyboard and then hit the key. (Note that even a novice typist has the skill to hit a key by moving a finger.) Only after this is finished will he proceed to find and hit the next key "y". Performance is serial and slow. Compare this with an expert who no longer has to work letter by letter, but can deal with several keystrokes simultaneously. In fact, skilled typists seem to achieve their speed by overlapping their finger movements for successive keystrokes (Gentner, Grudin & Conway, 1980). As the typists learn to type faster, their finger movement patterns change so as to take into account the context of each character (Gentner, 1983). Performance becomes parallel and fast.

The model to be presented here, called the HS model (for Hierarchical-Sequential Model), was designed to account for this kind of change from a novice to an expert. The HS model, like many other hierarchical models (Miller, Galanter & Pribram 1960; MacKay, 1982; Laird, Rosenbloom, & Newell 1986; Anderson, 1982, for example), assumes a hierarchical control structure, where higher level representation holds more abstract, longer range information, whereas more concrete, short term information is represented at lower levels. Thus, an intention to perform a sequence is represented at the top level and is converted to representations at successively lower levels until it is finally converted into actual physical body movements at the lowest level. For example, the highest level might specify an intention to type a sequence of letters (word, phrase, or sentence), and the intermediate levels might represent letter subsequences, or individual letters, that constitute the whole sequence. Consequently, higher level representation stays relatively stable whereas representations at lower levels changes more rapidly: an intention to type a word stays unchanged at the top level while the lowest level goes through a sequence of action components, eg., finger movements. The HS model differs, however, from the previous hierarchical models in that there is no fixed a priori relations between levels of representation and the levels in the hierarchy of the model, except at the highest and the lowest levels. The relations change as the system learns what level of information each level in the model should represent in order to achieve more efficient performance.

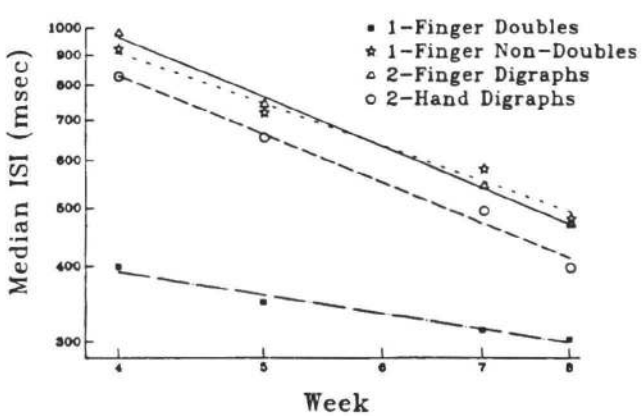

Figure 1. Learning curves for one of the typists studied by Gentner (1983). The median interstroke intervals are plotted against the number of weeks in a beginning typing class on a log-log scale. The four curves correspond to four different digraph classes that differ in their motoric requirements, but the specific effects of digraph classes are not important for the current discussion.

In this framework, the learning process from a novice to an expert described above can be characterized as follows: in the novice case, lower, more "motoric" levels have not developed representations of long sequences but have representations only of small action components, such as "hitting a key". This means that the representation of an intention must be broken down into smaller components at relatively higher levels in the hierarchy. However, as the system becomes expert, representations of chunks of these components (such as small letter sequences) are developed at intermediate levels. Such chunks are broken down into their constituents at lower levels, closer to physical movements. If we only assume that it takes a constant time to execute a chunk at an intermediate level, forming chunks of longer subsequences leads to a faster performance because fewer chunks are required to represent each sequence,

The Network Architecture

A special case of the HS model, in which there are three levels of representation, has been implemented and tested as a PDP network. Since the model was described fully in Miyata (1987, 1988), I will only briefly review the model and summarize the findings previously reported. In the next section, I will describe new findings about the model's learning process. Figure 2 shows the architecture of the network. The highest level, labeled *Intention*, contains a conceptual representation of the action sequence to be performed. The lowest level, labeled *Action* represents individual components to be executed. The middle level, *Plan* mediates the mapping between *Intention* and *Action*. The operation of the model involves two mappings, implemented by two subnetworks: The subnetwork Planning-Net maps from an *Intention* vector to a sequence of *Plan* vectors. The subnetwork Execution-Net maps from a *Plan* vector

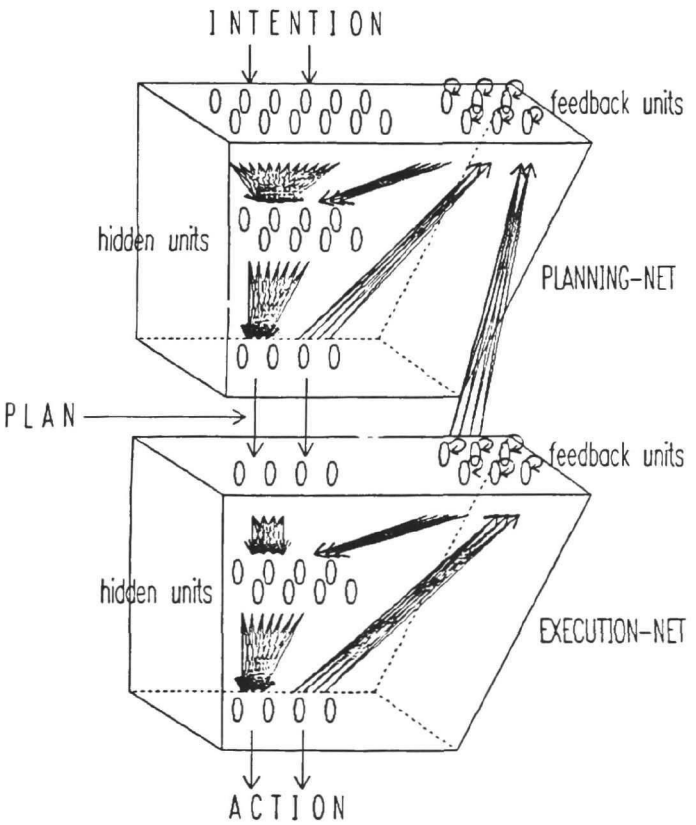

Figure 2. The architecture of the HS model with three levels of representation. *Intention* is a conceptual representation of the action sequence to be performed. *Action* represents individual action components to be executed. *Plan* is an intermediate representation that mediates the mapping between *Intention* and *Action*. A single *Intention* vector is mapped to a sequence of *Plan* vectors by Planning-Net, and each *Plan* vector is mapped to a sequence of *Action* vectors by Execution-Net. The two mappings are implemented by two Jordan networks. The output of Planning-Net is directly fed to the plan units of Execution-Net. The feedback units of the Execution-Net are connected to the feedback units of the Planning-Net.

to a sequence of *Action* vectors.

Each subnetwork was Jordan's sequential network (Jordan, 1986) in order to generate a sequence of output vectors from a single input vector. In addition to a feedforward three-layer architecture with one layer of hidden units, a Jordan network has a set of feedback units with recurrent self-connections which acts as a memory of a temporal context of past output vectors stored as an exponentially decaying trace of past output vectors: The sequence of vectors $X = (\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_T)$, where $\vec{x}_t$ is the output vector at time t, is stored as $\sum_{t=1}^{T} \alpha^{T-t} \vec{x}_t$, where α is the decay factor ($0<\alpha<1$). At each time step, the next output vector is determined both by its input vector (which does not change during the sequence), and by the feedback vector (which changes at each time step).

In the HS model, a set of connections from the feedback units of the Execution-Net to the feedback units of the Planning-Net allowed the latter to keep track of what the former was doing. Also, note that Planning-Net operated at a slower rate than Execution-Net: Planning-Net is updated only once in every three steps (in this particular simulation) of updating Execution-Net.

In the simulation reported here, there were 4 possible actions A, B, C and D, each represented by one of 4 output units of Execution-Net. Each *Intention* vector represented a sequence of three actions.

Pre-Training

The back-propagation algorithm (Rumelhart, Hinton, & Williams, 1986) was used to train the network. However, before the system could start learning, the elementary skill of a novice, such as the ability to find and hit a key, must be somehow realized in the system. This prior knowledge was modeled by pre-training the network so that it could perform in a manner analogous to a novice typist, before the actual training of the task itself started. As the result of pre-training, the network could perform the task but only slowly. Figure 3 illustrates the time course of the operation of the network after the pre-training. It was trained to use 4 *Plan* vectors, each representing one of the 4 possible actions. In order to generate the sequence *ABC*, for example, Planning-Net was pre-trained to generate a sequence of three *Plan* vectors, (*Plan*1, *Plan*2, and *Plan*3) one representing A, one for B, and one for C. Execution-Net was pre-trained to respond to each *Plan* by turning on the corresponding output unit (shown in the figure by the upright rectangle in the action sequence) at the first time step and then turn off all output for the next two time steps.[1]

The network was trained to produce the 64 possible sequences of three components, each component being one of four actions.

Training

In the actual training, a procedure was used that forced the network to gradually speed up its performance. Suppose the network was to produce the sequence *ABC*. Each action produced by the network was compared against a target and the weights modified so as to reduce the error. The target was always the next component in the sequence to be produced. Initially, the target was the first component *A*. The target stayed the same until the *Action* vector matches the target.[2] Thus, if the network generates a wrong action, e.g., *B*, or *C*, instead of *A*, the target continues to be *A*. When the action matches the target,

---

[1] The choice of the representational formats for the *Intention* vectors and for the initial *Plan* vectors are mostly arbitrary. In this simulation, a local representation, in which each unit represented a particular action at a particular point in time, was used to avoid any unwanted effects of similarity structures embedded in the representation.

[2] Which action the network has produced was decided by choosing the most active output unit.

however, the target is changed to the next component in the sequence, in this case, *B*. A property of this procedure is that the faster the sequence is produced, the smaller the overall error becomes. Figure 4 shows the response of the network generating the sequence *ABC* after 1600 presentations of all 64 patterns. Only one *Plan* was necessary to specify the sequence to Execution-Net, from which Execution-Net generated the whole sequence ABC. All other sequences were also completed with one *Plan* vector. Thus, the network developed a representational format that could encode all 64 sequences of three actions in *Plan* vector.

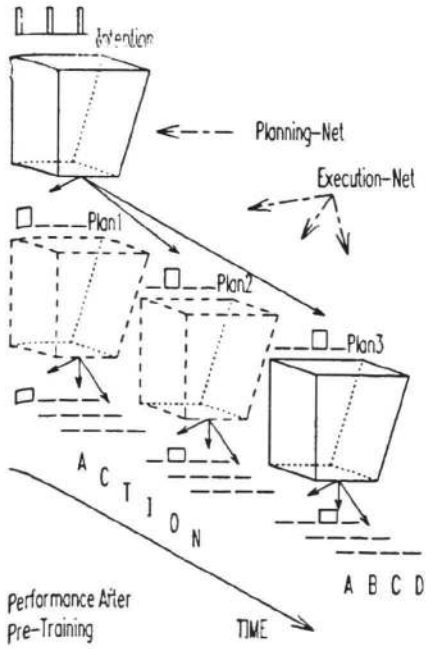

Figure 3. The time course of updating the state of the network. Planning-net maps from an *Intention* to a sequence of three *Plan* vectors. Execution-net maps from each *Plan* to a sequence of three *Output* vectors. The figure shows the response of the network after the pre-training phase. Execution-Net could generate only one action component from a *Plan*. In order to generate the sequence *ABC*, Planning-Net has to generate a sequence of three *Plan* vectors representing *A*, *B*, and *C*. After the pre-training, the network could produce all 64 sequences of three actions but only slowly. It takes seven time steps to complete each sequence.

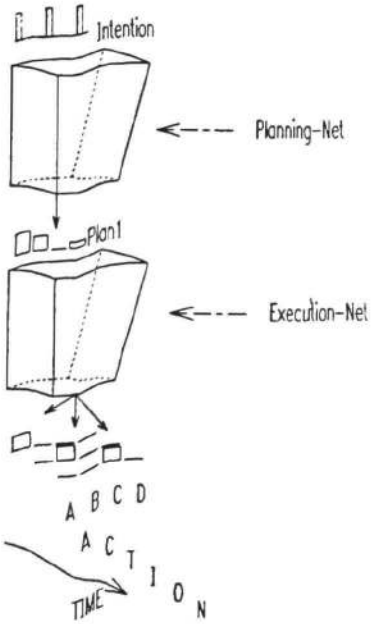

Figure 4. Response of the network after the training phase to the same input as in Figure 3. Only one *Plan* vector is needed to specify the sequence *ABC*. All 64 sequences were each represented by a single *Plan* vector and thus completed in three steps. The *Plan* units which, before the training, could represent only one action component at a time, have learned to represent the entire sequence. Before the training, Execution-Net did only a simple mapping of one *Plan* to one *Action*, and much of the work was done by Planning-Net that mapped an *Intention* to a sequence of *Plans*. The training reversed the situation: the mapping from *Intention* to *Plan* is now one-to-one, and the mapping from *Plan* to *Action* is one-to-sequence. From Planning-Net's viewpoint, the task has changed from a serial one to a highly parallel one.

ANALYSES AND DISCUSSION

The HS model was designed so as to model a certain characteristic of the change from a novice performance to an expert performance. When its learning process was examined more closely, it revealed some other interesting characteristics that are also known for human skill learning, some of which were reported in Miyata (1987). I will describe here some recent findings.

The Learning Curve

To obtain the learning curve, an HS network was tested with three possible actions: the network had three units each in the *Action*, and *Plan* layers and nine units in the *Intention* layer. The network was trained on all the possible sequences of three outputs, each output being one of three possible actions. There are 27 such sequences. Eight networks, with different initial random weights, were trained, and the duration (number of time steps) to generate each sequence was recorded during the training. Figure 5-(a) shows the learning curves, plotted in log-log coordinates, obtained for the eight networks. Each datum is obtained by averaging over a period of 10 training trials, each trial consisting of the 27 sequences. The straight line in each plot shows the best fit linear regression in the log-log space. The learning curves tend to lie along a straight line, and the deviations from the linearity do not seem to show any systematic pattern, except for the apparently asymptotic leveling at the end of some of the curves. This effect will be discussed below. In fact, when these curves were averaged (Figure 5-(b)), it yielded a very good fit to a straight line. (r^2=0.99 by averaging in the original raw data. A very similar result with r^2=0.97 when averaged in the log scale.) This suggests that the deviations from the linearity (in the log-log space) seen in each learning curve are not systematic across different networks. In human learning, the learning rate

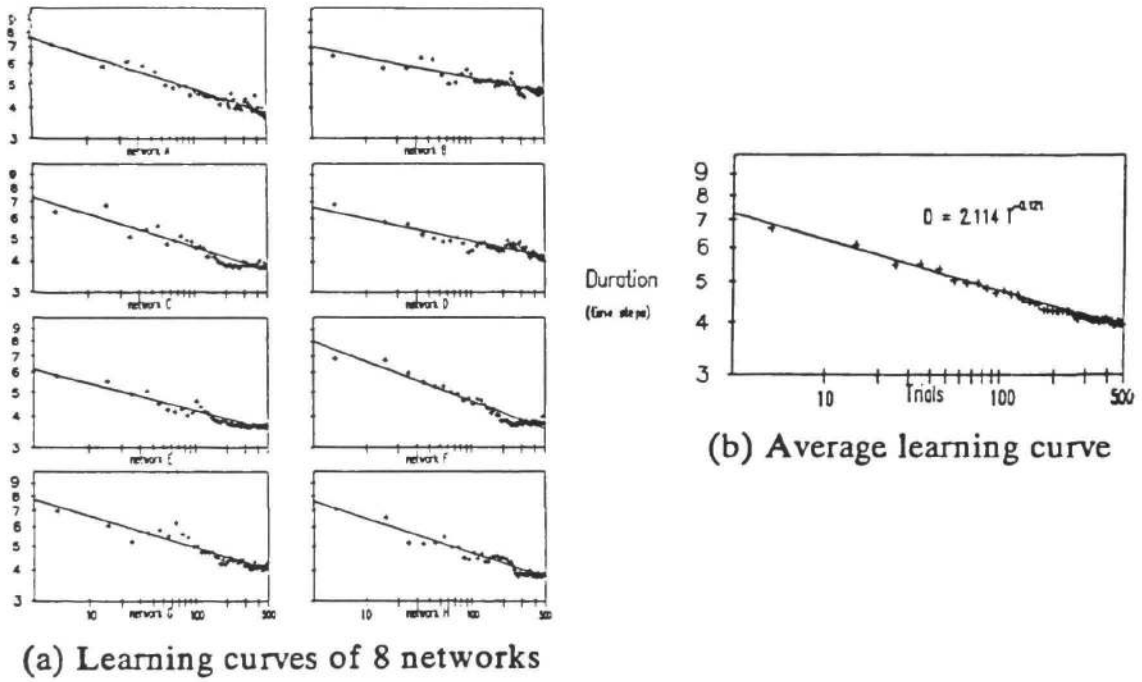

Figure 5. The learning curves in log-log coordinate for the eight networks (a), and the average learning curve (b). The best fit linear regression line is also shown.

(the slope of the line) has been found to vary with the task (Newell & Rosenbloom, 1981; MacKay, 1982), with individual subjects, and with different motoric components in typing (Gentner, 1983). For the 15 data sets analyzed by Newell and Rosenbloom, the value of the learning rate varied from 0.06 to 0.81. For the eight networks shown in Figure 5-(a), it ranged from 0.079 to 0.157.

Asymptotic deviation from linearity such as observed in some of the data in Figure 5-(b) were observed in many of the data sets examined by Newell and Rosenbloom (1981) and by MacKay (1982). The slope of the learning curve often diminished at the end; the beginning of the curve sometimes slightly deviated from linearity (usually downward). Gentner (1983) pointed out that the learning rate of the beginning typist shown in Figure 1 could not continue indefinitely: such typist would be typing at 370 words per minute after 4 year. The improvement must have some asymptotic level eventually.

When the learning rate parameter in the HS network which determines the magnitudes of weight changes in proportion to the errors was varied, the learning curve remained approximately linear for a wide range of parameter values. (The data presented here was obtained with the parameter value of 0.05.) For a very small learning parameter (below 0.01), however, the learning curve deviated downward from a straight line at the beginning.

The shape of the learning curve of the HS model can be understood, at least qualitatively, as follows. Note that in order to achieve a performance speed of kS_0, where S_0 is the initial speed, the *Plan* units have to learn to represent all N_a^k sequences of k primitive actions, where N_a is the number of the primitives (assuming an exponential environment where all combinations of the primitives must be learned, and uniform learning across sequences.) If we take as the measure of difficulty of learning the number of new subsequences to be learned by the *Plan* units in order to achieve a constant amount of speed up, we see that learning becomes exponentially more difficult as the performance becomes faster. In order to derive the learning curve, however, we need a better understanding of the behavior of the learning algorithm itself to relate this measure to the time it takes to learn.

Laird, Rosenbloom, and Newell (1986) showed that their Chunking Theory can account for the power law in a variety of tasks. Currently, the HS model deals with only one of three components in the framework of the Chunking Theory, namely the decoding process in which a representation of a response sequence is decoded into its constituents. Consequently, in order to apply the HS model to the wide range of tasks that the power law has been observed, it needs to be combined with models of the other two components, encoding of stimuli and connection between the encoding and decoding processes (for example, Miyata, 1988b; see Miyata 1988a, for a preliminary discussion).

Frequency effect on substitution errors

A strong effect of frequency on errors in typing has been found at the level of individual letters by Grudin (1983) who examined the confusion matrix (a table showing the frequency with which a letter is typed in place of another for every combination of letters) compiled by Lessenberry (1936) as well as his own data. When homologous errors (striking the key occupying the "mirror-image" position on the keyboard with respect to the correct one) and adjacent errors (striking a key adjacent to the correct one) were analyzed, it was found that higher-frequency letters were more likely to replace lower-frequency letters. When a Jordan network (a simplest case of the HS model) was trained to produce a set of sequences such that each component was presented with different frequency, its error patterns also showed a strong effect of frequency. For all 13 pairs of components with different frequencies, the probability of replacing the lower-frequency component with the high-frequency one was higher than the probability of replacing in the opposite direction.

CONCLUSION

I have described a PDP model of skill learning that readily accounted for the increase in speed and the shift from serial to parallel performance. The learning was achieved by incrementally modifying the mappings in the network so that the internal *Plan* units represented gradually longer subsequences. It is encouraging that the model has yielded, as emergent properties, a number of phenomena that are found in human skill learning. It remains to be studied what factors in the model and the task affect the learning curve, eg., its slope and deviation from linearity, and how. (One possibility is that the learning rate increases with the number of plan units used to represent the sequences, and decreases with the number of sequences that must be learned by the network.) Such study can be compared against subject's performance in similar situations.

REFERENCES

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*, 369-406.

Gentner, D. R., Grudin, J., & Conway, E. (May 1980). *Finger movements in transcription typing* (Report No. 8001). Institute for Cognitive Science, University of California San Diego.

Gentner, D. R. (1983). The acquisition of typewriting skill. *Acta Psychologica, 54*, 233-248.

Grudin, J. T. (1981). *The organization of serial order in typing.* Unpublished doctoral dissertation, University of California at San Diego.

Grudin, J. T., & Larochelle, S. (1982). *Digraph frequency effects in skilled typing* (Tech. Rep. No. 110). Center for Human Information Processing, University of California, San Diego.

Grudin, J. T. (1983). Error patterns in skilled and novice transcription typing. In W. E. Cooper (Ed.), *Cognitive aspects of skilled typewriting* (pp. 95-120). New York: Springer-Verlag.

Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the eighth annual conference of the Cognitive Science Society* (pp. 531-546).

Laird, J., Rosenbloom, P. & Newell, A. (1986). *Universal Subgoaling and Chunking: the automatic generation and learning of goal hierarchies.* Kluwer Academic Publishers.

MacKay, D. G. (1982). The problems of flexibility, fluency, and speed-accuracy trade-off in skilled behavior. *Psychological Review, 89*, 483-506.

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). *Plans and the structure of behavior.* New York: Holr, Rinehart, & Winston.

Miyata, Y. (1987). Organization of action sequences in motor learning: a connectionist approach. In *Proceedings of the ninth annual conference of the Cognitive Science Society* (pp. 496-507). Seattle, WA. Also, Tech. Rep. No. 8707, Institute for Cognitive Science, UC San Diego. La Jolla, CA.

Miyata, Y. (1988a). *The learning and planning of actions.* PhD Thesis, Psychology Department, also Tech. Rep. No. 8802, Institute for Cognitive Science, University of California, San Diego.

Miyata, Y. (1988b). An unsupervised PDP learning model for action planning. In *Proceedings of the tenth annual conference of the Cognitive Science Society* (pp. 223-229). Montreal.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive Skills and their Acquisition.* Hillsdale, NJ: Erlbaum.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representation by error propagation. In D. E. Rumelhart, J. L. McClelland (Ed.), *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1: Foundations.* Cambridge, MA: MIT Press/Bradford Books.

Sellen, A. J. (1986). *An experimental and theoretical investigation of human error in a typing task.* Unpublished Master's thesis, University of Toronto.