Organization of Action Sequences in Motor Learning: A Connectionist Approach

Yoshiro Miyata
Institute for Cognitive Science
University of California, San Diego

Abstract

This paper presents a connectionist model of motor learning in which performance becomes more and more efficient by "chunking" output sequences, organizing small action components into increasingly large structures. The model consists of two sequential networks: one that maps a stationary representation of an intention to a sequence of action specifications or action plans, and one that maps an action plan to a sequence of action components. As the network is trained to produce output sequences faster and faster, the units that represent the action plans gradually discover representational formats that can encode larger and larger chunks of subsequences. The model also shows digraph frequency effects similar to that observed in typewriting, and it generates capture errors similar to that observed in human actions.

Organization of Action Sequences

The idea that the size of perceptual and motor units increases with experience is not new. Various models have proposed different ways in which the units are organized: in a hierarchical manner (Bryan & Harter 1897, Lashley 1951); as associative chains of small elements (Wickelgren 1969); or elements linked together by inhibitory connections (Rumelhart & Norman 1982), for example. Recently, Grudin and Larochelle (1982) and Jordan (1986) have argued that when a complex motor skill such as typing is learned, the learner develops representations of motor sequences at levels higher than individual action components, such as digraphs in typing. Representation of "chunks" of motor sequences seems to be formed.

The connectionist framework (Feldman & Ballard 1982, Rumelhart & McClelland 1986) has been successfully applied to the domain of motor control in a number of studies (Hinton & Smolensky 1984, Rumelhart & Norman 1982, Jordan 1986). These studies stressed the role of parallel computation in solving the problems of many degrees of freedom and of multiple and complex constraints. However, the issues of how representation of action sequences is developed and how performance becomes increasingly efficient have not been addressed directly by these models. In general, these models tended to focus on the performance of an expert rather than the process of learning itself. This paper presents a model of motor learning, of the shift from a novice to an expert performance.

The Architecture of The Model

In the model, there are three basic levels of representation. The first is a conceptual representation of the sequence to be produced, such as a word to be typed, that is more or less independent of motoric components or physical requirements of the task. The second is a representation in which actions are specified to the action system. The third is the output of the action system that represents each component to be executed. The execution of an action sequence in the model thus involves two mappings: the mapping from the conceptual representation to the action specification, and the mapping from the action specification to the actual action components. In both mappings, an input vector is mapped to a sequence of output vectors.

Jordan's Network

One important requirement for a model of action control is that it is capable of generating sequences of output vectors. The simulations described in this paper used a technique developed by Jordan (1985) to generate sequences. Figure 1 shows the architecture of a Jordan network. It receives an input to a layer of *plan* units, called the *plan vector*. For example, a plan vector might specify the sequence "ABCD.." and the task of the network is to produce first output vector A, then B, C, and so on. Once a plan vector is

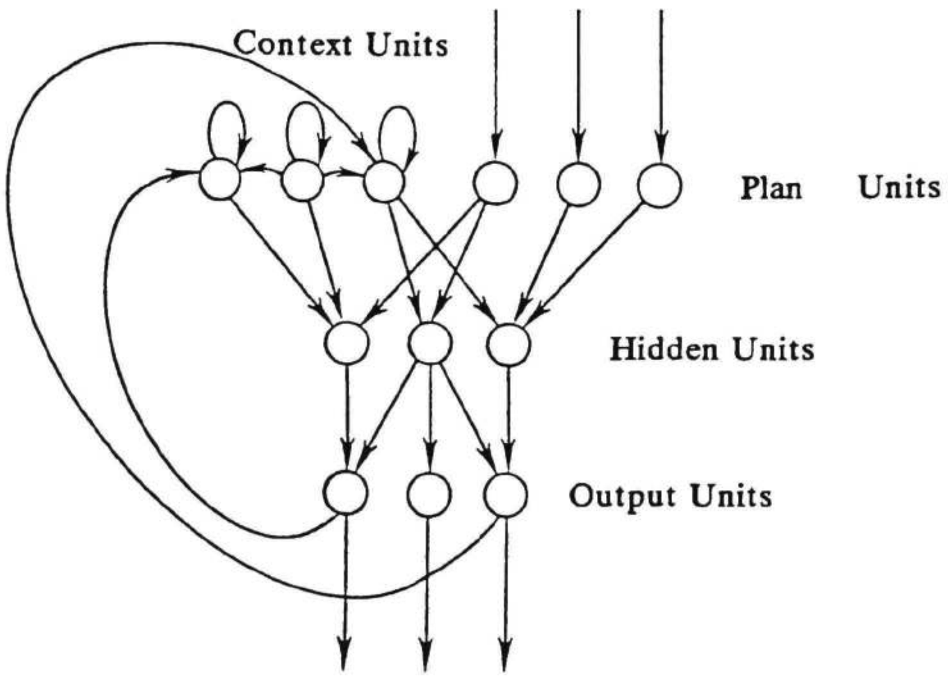

Figure 1. The basic architecture of a Jordan network. Output is determined by a stationary plan and temporal context of past outputs.

given, it produces the first output vector by sending activation forward through one layer of hidden units. Then the output vector is fed back to a layer of *context* units, which store the history of output vectors by recurrent connections to themselves. At each time step, the next output vector is determined both by the plan vector which does not change during the sequence and by the context vector which changes at each time step and thereby specifies place in the sequence.

The Hybrid Model

The model was constructed as a hybrid of two Jordan networks, one for each mapping (the left half of Figure 2). The upper network, called *Plan-net* supplies plan vectors for the lower network called *Action-net*. The right half of Figure 2 illustrates the time course of updating the state of the network. At time 0, a specification of the sequence to be produced is given at the layer of units labeled "Intention" which serves as the input to Plan-net. Plan-net then generates its output as a sequence of three vectors at the layer of units labeled "Plan" at time 0, 3, and 6. The plan units actually represent the input for Action-net. In response to each plan vector, Action-net generates a sequence of three output vectors, one at each time step, at the layer labeled "Output". Then each output vector is converted to another vector at the layer labeled "Action" by choosing the most active output unit. Thus, Action-net updates its state more often than (in this particular simulation, three times as often as) Plan-net. There is a connection from the context units of Action-net to the

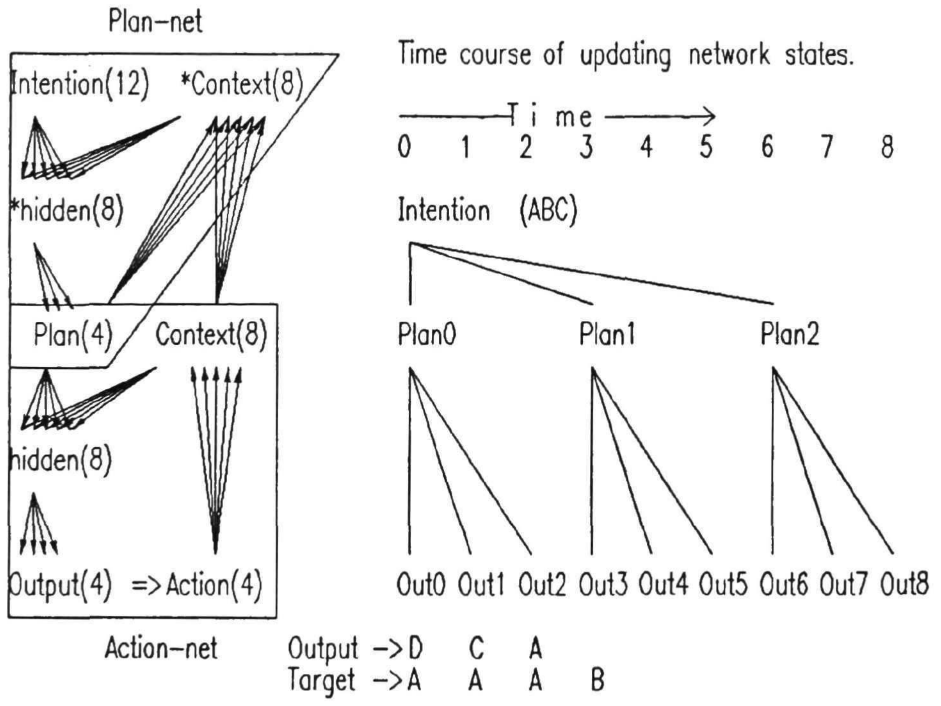

Figure 2. The architecture of the model (left) and the time course of updating the state of the network (right). Plan-net maps from an *Intention* to a sequence of three *Plans*. Action-net maps from each *Plan* to a sequence of three *Outputs*.

context units of Plan-net: this allows Plan-net to get some information about what Action-net has done so far.

Training The Network

The entire network was trained using the back-propagation algorithm (Rumelhart, Hinton & Williams 1986). The training procedure for a network with recurrent connections is slightly more complicated than the training of a straight feedforward network and is described in detail in Rumelhart et al., (1986).

Each output vector and each action vector were compared with a target vector. Initially the target vector represented the first component to be produced in the sequence. For example, suppose the target sequence is *ABC*. Then the target to compare with the output and action vectors is initially *A*. If the action vector does not match the target then the target stays the same so that the same target is used at the next time step. If the action vector does match the target then the target for the next time step is changed to the next component, *B* in this case. At each time step, the error between the output vector and the target is propagated back through the network and all the weights in the network[1] are modified so as to reduce the error.

Pre-training

In any learning situation for humans, the learner usually has a fair amount of a priori or background knowledge before the learning starts. This prior knowledge was modeled by dividing training into two parts: one to put background knowledge and one to train the task itself. Consider the typing task : Even a novice typist can type correctly by "hunt and peck." The problem is that the novice is very slow. The network was pre-trained so that it can perform the task analogous to the performance of a novice typist. Thus the Plan-net was trained to generate a sequence of plan vectors, each plan vector representing only one action component. The Action-net was pre-trained so that in response to each plan vector it could generate the action represented by the plan vector.

Figure 3 illustrates the performance of the network after the pre-training phase. An activity pattern in a layer of units (a vector) is shown as a row of vertical bars, the height of a bar representing the activation level of a unit. A sequence of vectors is shown as a matrix composed of several rows of vertical bars, each row representing the vector at each time step. The bottom row represents the first vector produced and the top row the last vector. The 12-dimensional vector labeled "Intention", is the input pattern to Plan-net at Intention layer. This pattern represents the output sequence *ABC*. In response to this input, Plan-net produces a sequence of three 4-dimensional plan vectors, shown to the right as three rows of vectors, representing the actions *A*, *B*, and *C*, respectively. In response to the first plan

[1] The recurrent connection from the units in Context layer to themselves had fixed set of weights which formed a diagonal matrix: i.e., each unit in Context layer was connected only to itself.

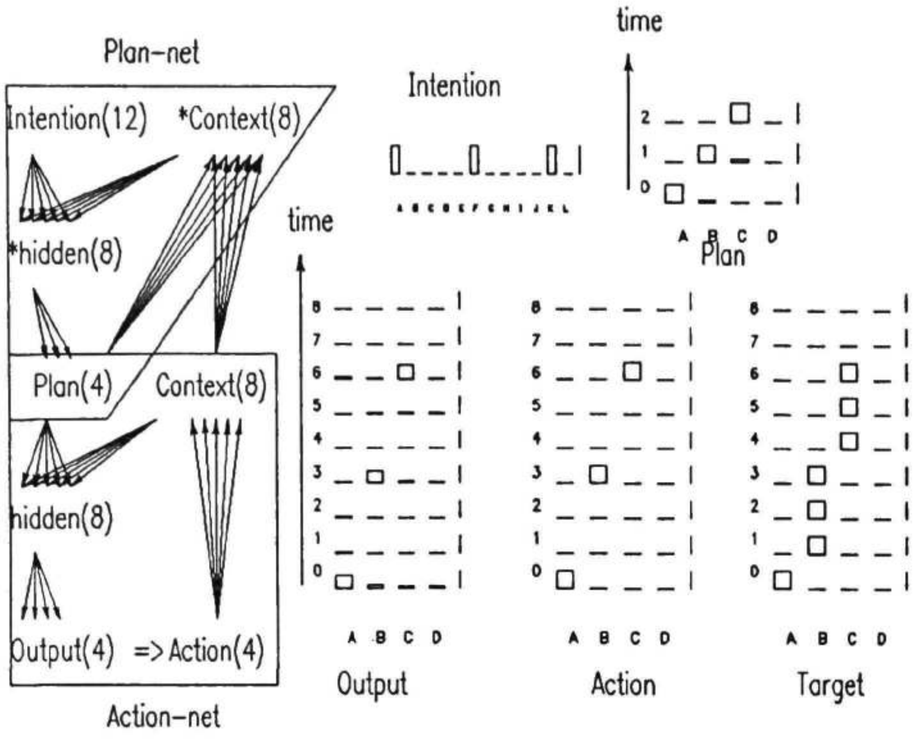

Figure 3. Response of the network after the pre-training phase: In response to an input (Intention) specifying the sequence *ABC*, Plan-net produces a sequence of three plan vectors (Plan) specifying the actions *A*, *B* and *C*. In response to each plan vector, Action-net produces the action specified by the plan. It takes seven time steps to complete the sequence.

vector, Action-net then produces the action *A*, and in response to the second plan vector it produces the action *B*, and so on. The three matrices at the bottom show sequences of nine output vectors, action vectors, and target vectors, at time step 0 through 8. In this simulation the network was trained on all the possible sequences of three outputs, each output representing one of four actions, *A*, *B*, *C* and *D*. There are 64 such sequences. After the pre-training the network can perform all the sequences but only very slowly.

Results - New Plan Representations

This situation changes as the result of training. Notice that as soon as the first action is made correctly the target shifts to the second component and tries to turn on the unit that corresponds to that component. In effect, this kind of training would be expected to speed up the execution of the entire sequence. That is in fact what happens. Figure 4 shows the response of the network to the same input sequence as shown in Figure 3 after some 1600 presentations of all 64 patterns. Before training, it took seven time steps to complete each sequence. After training, all the sequences are completed in three steps, which is the maximum rate. The plan units which, before the training, could represent only one action component at a time, now represent the entire sequence. These units have developed representational formats that can encode all 64 sequences.

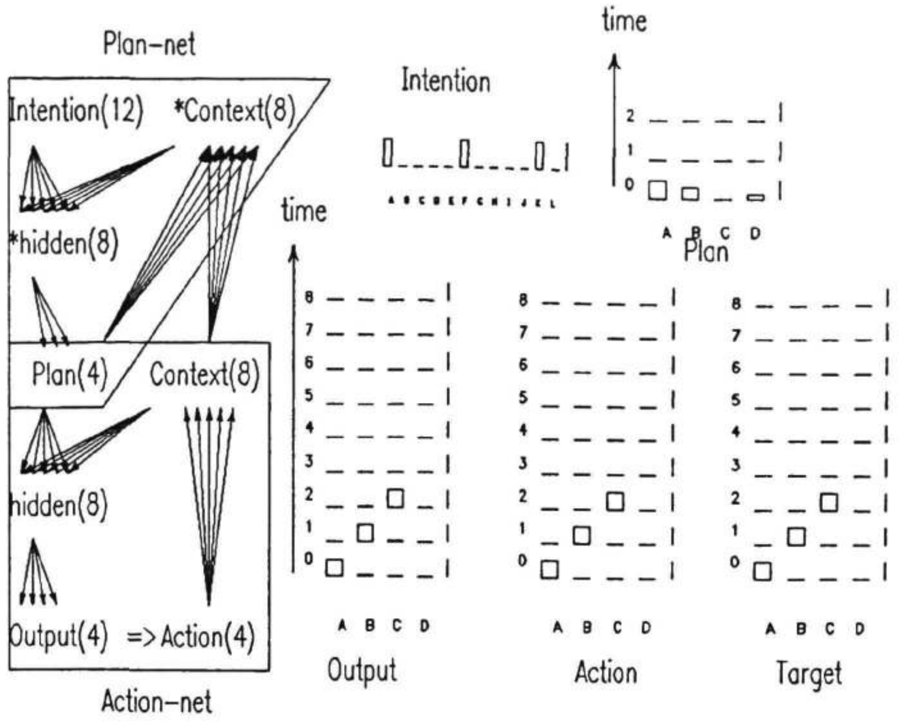

Figure 4. Response of the network after the training phase to the same input as in Figure 3. Only one plan vector is needed to specify the sequence *ABC*. The sequence is completed in three time steps.

The network, as the result of training, showed a transition from an inefficient performance analogous to that of a novice to a more efficient performance. The next section demonstrates that during the process of training, the performance of the network exhibits a certain characteristic also observed in human experts.

Digraph Frequency Effects

One strong source of evidence for the existence of higher-order (multi-character) representational units in typewriting comes from the digraph frequency effects. Grudin and Larochelle (1982) found that the inter-keystroke-intervals for higher frequency digraphs were reliably shorter than for lower frequency digraphs. The present model speeds up its output by developing higher-order representation of the output sequences. It is interesting to see if the model exhibits the same kind of effect: Does a transition from one action to another become faster if that particular transition is experienced more frequently by the network? To test this possibility, a set of twelve sequences of three actions were constructed using six digraphs (Table 1). Each sequence was presented with different frequency, as shown in the table, so that as a result three of the six digraphs (*AC*, *BA*, and *CB*) were presented twice as often as the other three (*AB*, *BC* and *CA*). A frequency means the number of times an item is presented to the network during each cycle of training. The network used in this simulation was identical to the one described in the previous section except that, because there were only three possible actions, there were only three units each in the output,

Table 1

Twelve sequences were presented to the network with different frequencies so that three of the digraphs contained were presented twice as often as the other three.

Frequency	Sequences presented to the network	Frequency	Digraphs contained in the sequence
4	ACB, BAC, CBA	High (8)	AC, BA, CB
2	ABA, ACA, BAB, BCB, CAC, CBC	Low (4)	AB, BC, CA
1	ABC, BCA, CAB		

action, and plan layers and nine units in the intention layer. The training procedure was identical to the one used before.

Results

Intervals (number of time steps) between two successive actions were recorded during the training phase. Three pairs of digraphs were compared: *AB* vs. *AC*, *BC* vs. *BA*, and *CA* vs. *CB*, the second digraph in each pair having the higher frequency. These pairs were chosen because two digraphs in each pair appeared in very similar contexts during the training: for example, *AB* appeared in the sequences *ABA, ABC, BAB* and *CAB* while *AC* appeared in *ACA, ACB, BAC* and *CAC*.

The network was faster with the higher-frequency digraph than with the lower-frequency digraph within every pair of digraphs. Table 2 shows the digraph frequency effects in each pair of digraphs during five blocks of 100 learning cycles. Following Grudin and Larochelle (1982), the "digraph frequency effect" (DFE) is defined as the average interval for the lower-frequency digraph minus the average interval for the higher-frequency digraph. This represents the time saved producing the second action of a higher-frequency digraph. The interval for a digraph is defined as the time steps required between the first and the

Table 2

Digraph Frequency Effects (time saved producing second action of higher-frequency digraph)

Blocks of learning cycles	Digraph Pairs Compared		
	AB—AC	BC—BA	CA—CB
500-599	1.27	2.18	1.37
600-699	1.36	1.80	1.43
700-799	0.81	2.20	1.43
800-899	1.00	2.00	0.86
900-999	1.04	2.04	0.51

second actions in the digraph. If the action *B* in the digraph *AB* is produced at time step immediately following the time step the action *A* was produced, then the interval for the digraph is 1. The overall digraph frequency effect was significant (i.e., DFE > 0), $F(1, 2) = 20.72, p < 0.05$, using the pairs as the random variable.

Capture Errors

One interesting aspect of action control is that errors seem to exhibit certain general patterns. This has been shown in controlled experiments in speech (Motley, Camden & Baars 1982, Dell 1984) and in typing (Grudin 1983, Sellen 1986) as well as observations in natural settings (Norman 1982, Reason 1979). One class of errors are called *capture errors*. "A capture error occurs when a familiar habit substitutes itself for an intended action sequence. .. Pass too near a well-formed habit and it will capture your behavior (Norman 1982)." The present model seems to have a potential to generate this type of errors. This is because the mappings in the network have the characteristic that similar inputs tend to be mapped to similar outputs. If two output sequences have similar subsequences, this results in similar contextual information in the context vector that might lead to an error.

This possibility was tested by simulating Sellen's psychological experiment (1986) using a Jordan network as the subject. The design of the experiment is summarized in Table 3. The network was given four plan-target pairs to learn. In response to each of the four plan vectors *a, b, c, and d*, the network was to generate four different output sequences *ABC, ABD, EFG, and EHI*. The first two sequences — *ABC* and *ABD* — are very similar to each other: these are called "high similarity sequences". The last two sequences — *EFG* and *EHI* — have only one common component: these are called "low similarity sequences". The sequences *ABC* and *EFG* are called "high familiarity sequences" because they are presented to the network three times more often than the "low familiarity sequences" *ABD* and *EHI*. There were nine possible actions, *A, B, C, ..., H, I*, and thus the network had nine output units. The action produced by the network was decided by choosing the most active output unit.

Table 3

Similarity and familiarity of the four plan-target sequence pairs learned by the network.

Plan -->	Target Sequence	Similarity	Familiarity
a	ABC	High	High
b	ABD	High	Low
c	EFG	Low	High
d	EHI	Low	Low

Table 4 shows the four possible "capture errors" that can occur. A capture error can occur either between two high similarity sequences ABC and ABC, or between two low similarity sequences *EFG* and *EHI*. It can occur either from a high familiarity sequence to a low familiarity sequence or vice versa. For example, if the network generates the sequence *ABD* in response to the plan 'a', it is a capture error between high similarity sequences and it is a high-to-low familiarity error.

The network used was a Jordan network with four plan units, six hidden units, four context units and nine output units. Each output unit represented one of the nine possible actions: *A, B, C,..., I*. The network was trained on the four pairs of plans and output sequences described above. Gaussian noise was added to each plan unit so that the network continued to make some errors even after it learned the sequences. Testing was done by presenting each plan, with noise added, one thousand times after the network had reached stable state. Seven networks were run that were identical except for initial random weights.

Results

Figure 5 shows the results. Both similarity and familiarity affected the probability of capture errors. There were more capture errors between the high similarity sequences *ABC* and *ABD* than between the low similarity sequences *EFG* and *EHI*, $F(1, 6) = 141.9, p < .001$. And there were more capture errors from a low-familiarity sequence to a high-familiarity sequence than from high to low familiarity, $F(1, 6) = 293.9, p < .001$.

In Sellen's (1986) experiment the familiarity factor had a significant effect on the probability of capture errors, but the effect of similarity was small and not significant. The discrepancy between her experimental results and this simulation can be attributed to a number of factors: The difference can simply be because the experiment did not have enough power to detect the effect (fewer trials, smaller overall error rate). Another possibility is that the temporal context stored in the context units did not correspond to the experiment. A finer grained manipulation of similarity in the experiment as well as manipulation of temporal characteristic of the context units in the simulation might be needed to make them

Table 4

Four possible capture errors

Plan ->	Target	Capture Error	Similarity	Familiarity
a	ABC	ABD	High	High -> Low
b	ABD	ABC	High	Low -> High
c	EFG	EHI	Low	High -> Low
d	EHI	EFG	Low	Low -> High

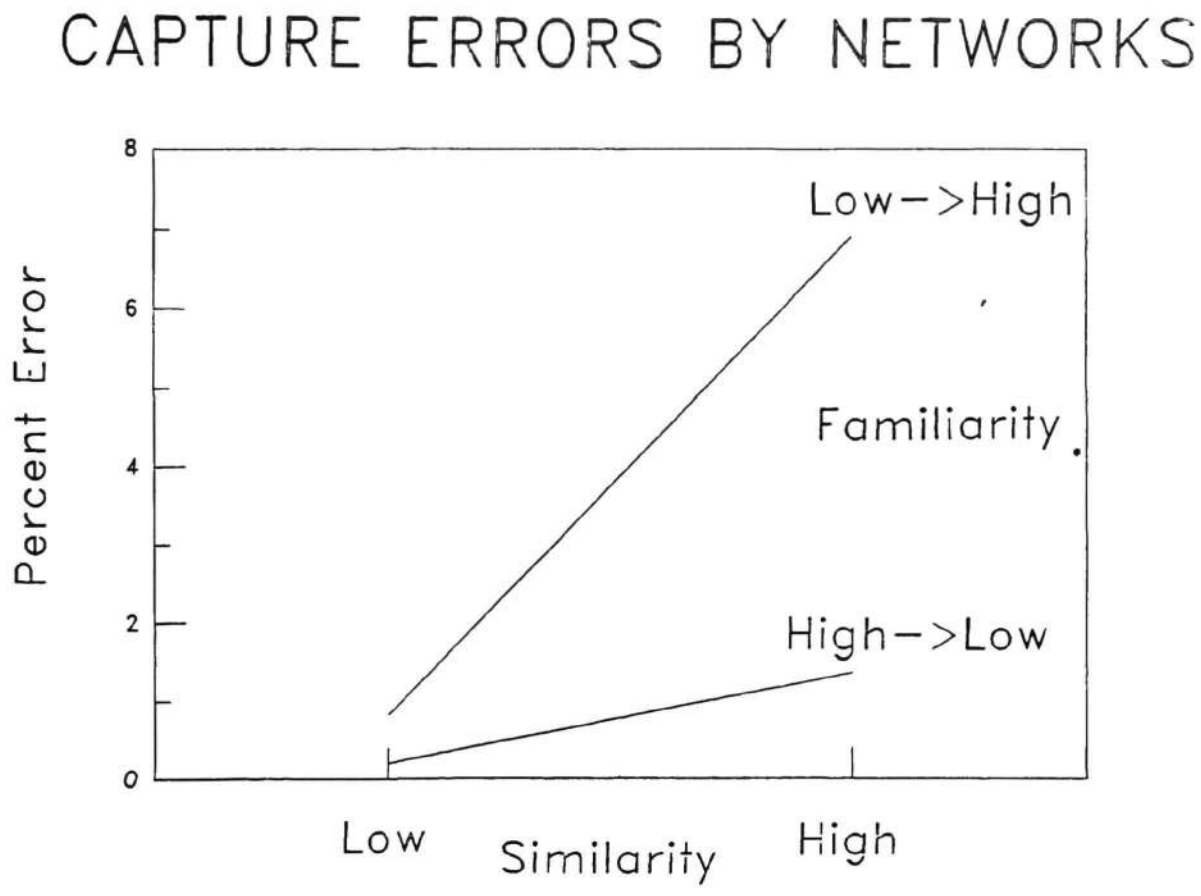

Figure 5. The effect of similarity and familiarity on capture errors made by seven networks: Capture errors were more likely from low-familiarity sequence to high-familiarity sequence than the reverse direction, and more likely between more similar sequences.

more comparable.

This simulation used a single Jordan network with noise to generate capture errors. Interestingly, capture errors were also observed without noise when the hybrid network learned many sequences. When some of the sequences were presented more often than other sequences, it was observed that a low-frequency sequence that shared a subsequence with a high-frequency sequence was sometimes captured by the high-frequency sequence, but a capture error in the reverse direction was rare. This suggests that some action errors can occur as the result of interaction among many different mappings that a single network is trying to learn.

Summary

The model presented in this paper has a number of interesting properties. First, it was able to model the shift from a serial, novice performance to a highly parallel, expert performance. Training of the network started from an initial state where Action-net did only a simple mapping of one plan to one action, and much of the work was done by Plan-net that mapped an intention to a sequence of plans. The training reversed the situation:

the mapping from intention to plan became one-to-one, and the mapping from plan to action became one-to-sequence. For Plan-net, the task changed from a serial one to a highly parallel one.

Second, this model seems to show a way "chunking" can be done in a connectionist network. The network was able to discover new representation of action plans so that each plan represents a increasingly large chunk of output sequence. The process of chunking in the model is gradual rather than discrete, and no new representational entity needs be created when chunks are formed. It is not necessary to presuppose what "level" of information each component in the model should represent: rather such a "level" is a dynamic property of each component that can change through learning.

Third, the model exhibits digraph frequency effects similar to those observed in studies of typing. The model also simulated capture errors and has the property that capture errors are more likely to be made from a low frequency sequence to a higher frequency sequence than the reverse direction.

Acknowledgement

This work was supported by a Grant from Nippon Telegraph & Telephone Corporation to the Institute for Cognitive Science. I would like to thank Don Norman, Dave Rumelhart, and Mike Jordan for useful discussions.

References

Bryan, W. L., & Harter, N. (1897). Studies on the physiology and psychology of the telegraph language: The acquisition of a hierarchy of habits. *Psychological Review, 4*, 27-53.

Dell, G. S. (1984). Representation of serial order in speech: Evidence from the repeated phoneme effect in speech errors. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 10*, 222-233.

Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science, 6*, 205-254.

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

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.

Hinton, G. E., & Smolensky, P. (1984). *Parallel computation and the mass-sprint model of motor control.* (Tech. Rep. No. 123). University of California, San Diego, Center for Human

Information Processing.

Jordan, M. I. (1985). *The learning of representations for sequential performance*. Unpublished doctoral dissertation, University of California, San Diego.

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).

Lashley, K. S. (1951). The problem of serial order in behavior. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior*. New York: Wiley.

McClelland, J. L., & Rumelhart, D. E. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 2. Psychological and biological models*. Cambridge, MA: MIT Press/Bradford Books.

Motley, M. T., Camden, C. T., & Baars, B. J. (1982). Covert formulation and editing of anomalies in speech production: Evidence from experimentally elicited slips of the tongue. *Journal of Verbal Learning and Verbal Behavior*, *21*, 578-594.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review*, *88*, 1-15.

Reason, J. T. (1979). Actions not as planned. In G. Underwood & R. Stevens (Eds.), *Aspects of consciousness*. London: Academic Press.

Rumelhart, D. E., & Norman, D. A. (1982). Simulating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive Science*, *6*, 1-36.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1. Foundations* (pp. 318-362). Cambridge, MA: MIT Press/Bradford Books.

Rumelhart, D. E., & McClelland, J. L. (1986). *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.

Wickelgren, W. A. (1969). Context-sensitive coding, associative memory, and serial order in (speech) behavior. *Psychological Review*, *76*, 1-15.