An Unsupervised PDP Learning Model for Action Planning

Yoshiro Miyata
Bell Communications Research

Whenever we have in mind something that we wish to bring about in the environment, we must find an appropriate sequence of actions that will result in the desired state. For example, if we decide to move from one place to another, we need to find an appropriate path between the two places and an appropriate sequence of actions. Likewise, when we reach for an object we must find an arm configuration such that the arm's tip touches the object as well as a set of muscle contractions that will result in that configuration. These are problems we encounter and solve with ease hundreds of times every day. What is the mechanism that enables us to map from the representation of a goal to the representation of the action plan for realizing that goal, and how can such capability be learned from experience?

Much work has focussed on *execution* of actions, namely, how a representation of action is converted into the right sequence of actions (for example, Rumelhart & Norman, 1982; Mackay, 1982; Rosenbaum, Hindorff & Munro, 1987). Not much work has been done to understand *planning* of actions, namely, how to find an appropriate action sequence to achieve an environmental state. Works in this domain have tended to require hand-wiring of task specific structures into the system and thus are not readily applicable to more general situations (for example, Hinton & Smolensky 1984; Anzai, 1984). Recent development of learning algorithms for Parallel Distributed Processing (PDP) networks (Rumelhart & McClelland, 1986), especially the back-propagation (BP) algorithm (Rumelhart, Hinton, & Williams, 1986), enables a network to learn task specific structures based on general principles. There have been extentions of the BP algorithm to sequential action execution (Jordan 1986; Miyata 1987) which have shown that the networks exhibit a number of characteristics observed in human actions (Miyata 1988). However, as discussed below, planning of actions requires more than learning a single input/output mapping, which the BP algorithm is designed to do.

THE COMPUTATIONAL REQUIREMENTS

Figure 1 illustrates the computational requirements of the situations that are considered in this paper. The process starts from $E_{desired}$, a representation of some desired environmental state. The system generates a representation of an action plan, A_{plan}, which is then executed (A). One requirement is that there is no teacher that gives the system the desired actions. The feedback provided to the system for

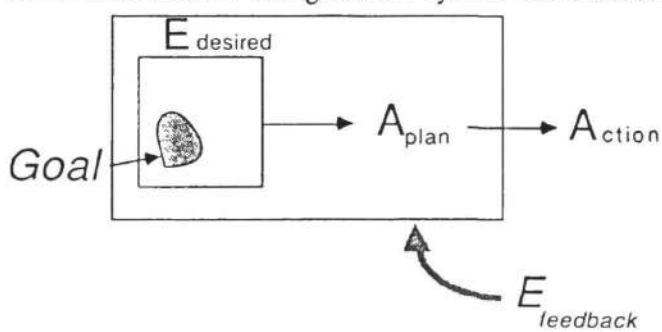

Figure 1. The computational requirements for the framework: (1) Feedback for learning is the environmental state as the result of an action plan; (2) A goal only partially specifies the desired environmental state. These requirements are not easily handled by the single mapping learning scheme.

learning is the environmental state as the result of executing the actions, $E_{feedback}$ (the *environment as the teacher* requirement). Another requirement is that $E_{desired}$ may be only partially specified by a goal of the task. For example, in a ship navigation task, the desired trajectory is usually only partially constrained by, say a gate or a channel (the *goal as partial specification of environment* requirement). Stated more precisely, a single goal specifies only some of the dimensions necessary to represent all possible goals. People seem to be able to learn and plan actions in situations defined by these requirements (see Anzai, 1984).

A common learning scheme is to characterize a task as a mapping from one representational space to another, and to train a network by presenting pairs of vectors from the input and the output space. One difficulty with applying this scheme to action planning is that the feedback provided to the learner from the environment does not necessarily specify what actions (outputs) the learner should have produced. Another difficulty is that, in order to generate an output, the input to a mapping must always be specified completely. This conflicts with the *goal as partial specification of environment* requirement. This paper presents one approach to these problems and proposes a framework in which the task-specific structure is learned as multiple mappings and planning is accomplished via interaction of these mappings that incorporates a constraint satisfaction process.

THE EAE NETWORK

The basic structure of the network, called the EAE (Environment-Action-Environment) network, is shown in Figure 2-(A). There is a set of units, called the E-units, for representing an environmental state, and another set, called the A-units, for representing an action plan. There are two mappings in the network: $E \rightarrow A$ mapping is the mapping from the E-units to the A-units; and $A \rightarrow E$ mapping is the mapping from the A-units to the E-units. Each mapping is implemented through a layer of hidden units.

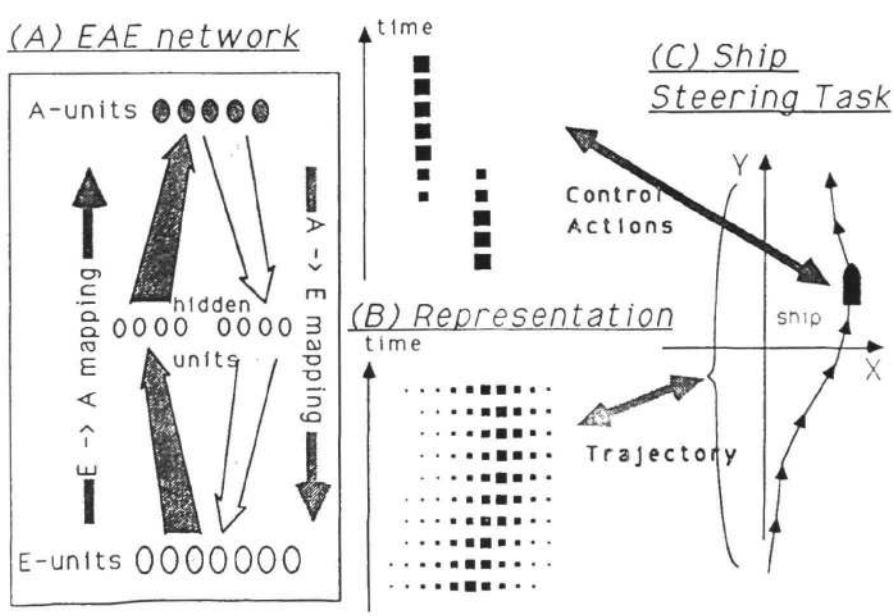

Figure 2. (A) The basic structure of the EAE network. E-units represent an environmental state and A-units an action plan. The $E \rightarrow A$ mapping maps from the E-units to the A-units, and the $A \rightarrow E$ mapping from the A-units to the E-units, each through a set of hidden units. (C) The network was applied to a ship steering task. The ship moves at a constant speed along the Y-axis of a 2-dimensional space and the network controls the acceleration of the ship along the X-axis. (B) The two matrices of squares show an activation pattern in the E-units representating a trajectory of the ship and an activation pattern in the A-units representing an action plan.

The EAE network has been applied to a task of navigating a ship in a 2-dimensional space (Figure 2-(C)), similar to an experimental task used by Anzai (1984). The ship moves at a constant speed along the Y-axis, and the network's action controls the acceleration along the X-axis. The E-units represent a trajectory of the ship, and the A-units a sequence of actions for controlling the ship. A trial consists of 10 time steps. A trajectory for a trial, shown on the right, is represented as the activation pattern in the (110) E-units shown as an 10×11 matrix of squares (Firuge 2-(B)). Each row of 11 units represents the X-position of the ship at a point in time by a coarse coding. The pattern in the (20) A-units, shown as an 10×2 matrix, represents an action plan for a trial. A pair of units in each row represents the control value at a point in time. The ship's acceleration to the right is proportional to the activation of the right unit minus the activation of the left unit. In addition, there is a set of units (not shown in the figure) for representing the context, i.e., the ship's initial position and velocity. These units are fully connected to both hidden layers.

The network learns by executing many quasi-random actions and observing what trajectories are generated. The $E \rightarrow A$ mapping receives the trajectory as the input and uses the action as the target. (In other words, it learns the knowledge of the form "to get this trajectory do this action".) The $A \rightarrow E$ mapping learns using the same action/trajectory pair, but the action as the input, and the trajectory as the target. ("If I do this action then this trajectory results.") Both mappings are learned only by interacting with the environment.

After the mappings in the network were trained to some criterion, the network was given various kinds of goals and it was able to find an appropriate action plan for each of the goals. The planning process starts from a specification of a goal. A goal in this task is some part of the trajectory that the ship must follow, such as a gate or a channel. Figure 3 illustrates an example goal which is a channel that the ship must be steered into. This goal is specified by the activation pattern shown in the middle, in which the units representing a part of the trajectory are given activation values representing the goal. The activation of other E-units are not constrained by the goal, and are given some default initial value, in this case, all zeros. When given such a goal, the network tries to fill in the unspecified part of the trajectory as well as an action sequence that will result in that trajectory. The planning process is achieved by the following steps (see Figure 4):

1 A goal is set by clamping some E-units and zero activation in other units.
2 From this pattern, the $E \rightarrow A$ mapping generates an action plan in the A-units. If a complete trajectory is specified in the E-units, this mapping can generate an appropriate action plan to achieve the trajectory. However, because the trajectory is only partially specified, the generated plan is unlikely to be appropriate.

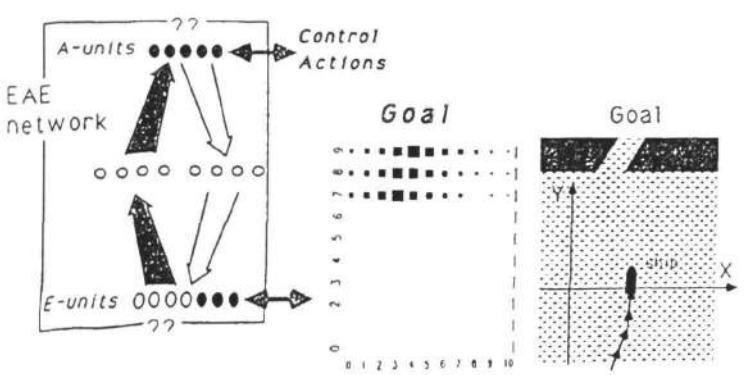

Figure 3. A goal is given by specifying activation values of some E-units representing partial trajectory, and the network tries to find activation patterns in the other E-units representing the rest of the trajectory, as well as an activation pattern in the A-units representing an action plan for achieving the trajectory.

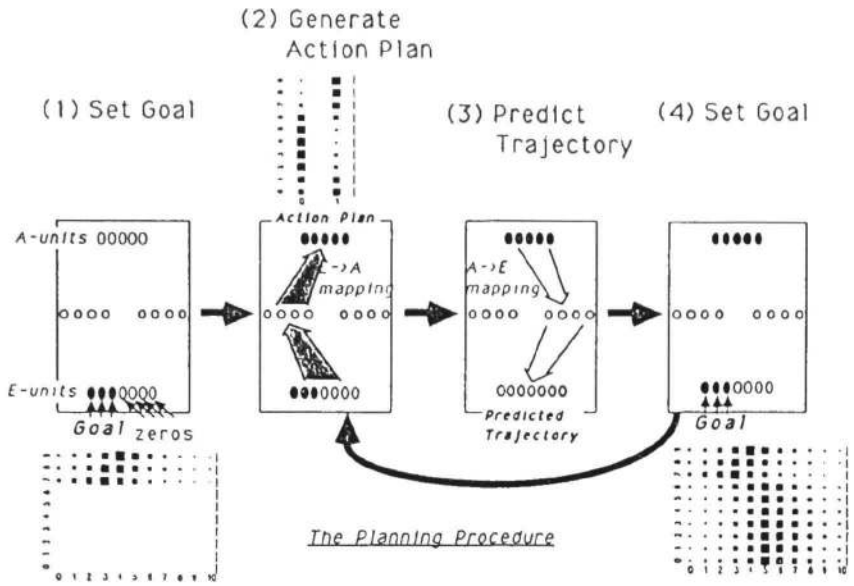

Figure 4. After the goal is specified in the E-units, the network maps back and forth between the E-units and A-units until the patterns settle into a stable state.

3 The $A \rightarrow E$ mapping generates, from the action plan, a predicted trajectory as the result of executing the plan.
4 This predicted trajectory may not satisfy the constraint of the goal. So, the E-units representing the goal are given the original values again.
5 The steps 2, 3, and 4 are repeated until the patterns settle into a stable state.

Figure 5 shows the series of patterns in the network during the process of planning after 1, 3, and 5 cycles. The leftmost column (**E**) shows the patterns in the E-units, after each time the goal is set. The second column (**A**) shows the patterns in the A-units. The third column (**Action**) shows the actual control values represented by these patterns. The rightmost column (**Trajectory**) shows what the trajectories would have looked like, if the actions were actually executed. As can be seen, the network was able to gradually improve the action plan. In this example, the patterns were stable after 5 cycles. This is one of

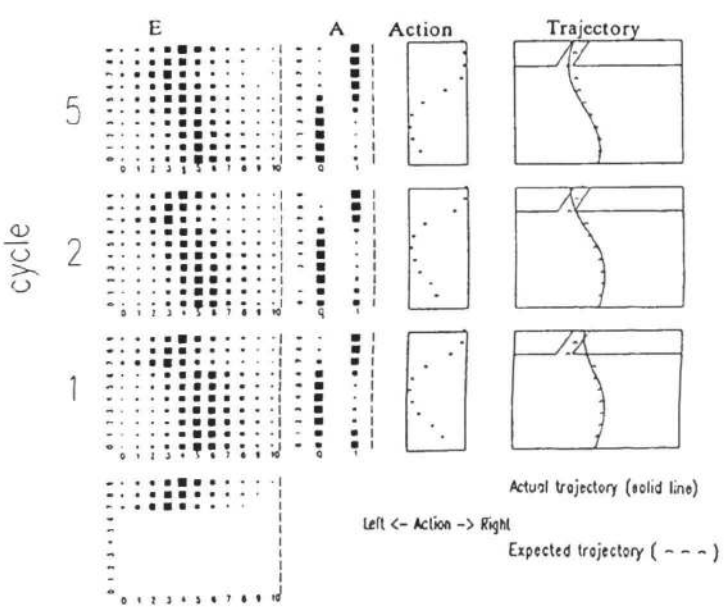

Figure 5. The activation patterns in the E-units (**E**) and the A-units (**A**) during the process of planning, after 1, 3, and 5 cycles. These patterns represent the trajectories and actions shown (**Trajectory**, and **Action**). The process starts from the representation of the goal in the E-units (bottom left) and the patterns in the E-units and in the A-units are iteratively mapped to each other until they are stable (cycle 5).

the most difficult cases, because a slight adjustment in the final direction of the ship requires a large change in the initial portion of the trajectory. Figure 6 shows the trajectories found by the network for 9 different goals and 3 different initial conditions.

ANALYSIS

Next, I present an analysis of the planning procedure and show that, under a number of assumptions, the process is expected to find an appropriate action plan. First, I make four assumptions. In figure 7, E-space is the space of all the possible activation patterns that can occur in the E-units. A-space is the space of all the possible activation patterns that can occur in the A-units. The first assumption is that all points in A-space are possible, i.e., they correspond to actual physical actions. This is reasonable because these patterns *cause* the physical actions. The second assumption is that only some points in E-space represent possible physical environmental states. This is reasonable because some physical states cannot be achieved by any action. Furthermore, some patterns may not correspond to any physical state. The third assumption is that the system learns the two mappings perfectly. The fourth assumption is that these mappings generalize to new patterns based on similarities to the learned patterns.

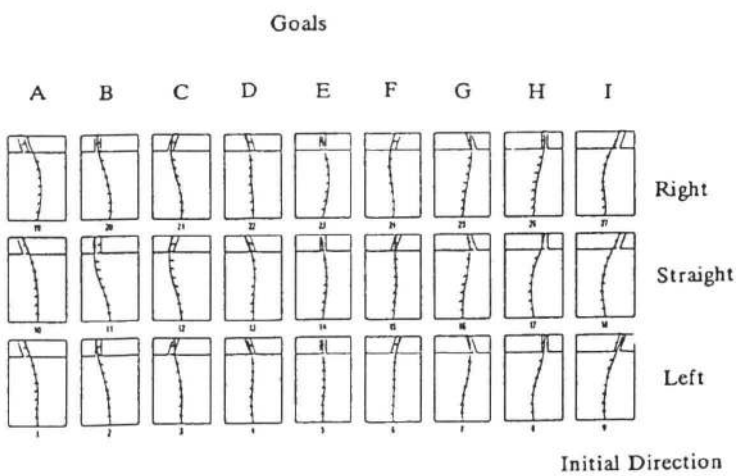

Figure 6. The trajectories found by the network for 9 different goals (A, B, .., I) and 3 different initial conditions (Right, Straight, Left).

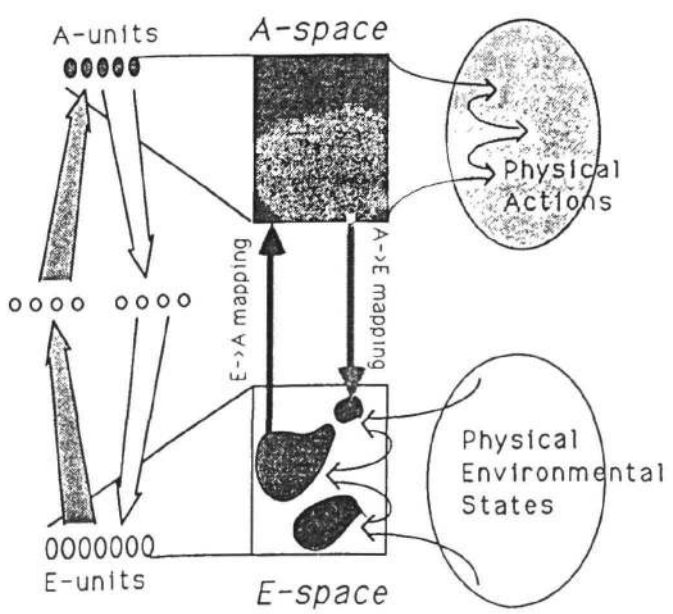

Figure 7. The four assumptions for the planning process: (1) All points in A-space represent some physical actions; (2) Not all points in E-space represent physical states; (3) Perfect $E \rightarrow A$ and $A \rightarrow E$ mappings; (4) Generalization based on similarity.

Such property has been demonstrated in many PDP networks (e.g., Cottrell, Munro, & Zipser, 1987; Chauvin, 1987).

Under these assumptions, the planning process can be analyzed as follows. (See Figure 8.) In E-space, the E_p regions are the points that represent possible physical states. For any particular goal, some dimensions of E-space are constrained by the goal. These dimensions, represented by the horizontal axis in the figure, correspond to the E-units that are given specific values by the goal. The other dimensions are not constrained by the goal. Thus, the goal defines a hyperplane in the space, called the E_c plane, which is parpendicular to all the constrained dimensions. The task of the planning is to find a *possible* state, a state that can be achieved by some action, that also satisfies the constraint of the goal. Such a state is represented by a point in an E_p region that is also on the E_c plane.

The EAE network searches for such a point by mapping back and forth between the two spaces. The network starts from $e_c^{(0)}$, a point in the E_c plane with zero activation values for the unconstrained dimensions. This point is mapped by $E \rightarrow A$ mapping to a point in A-space, $a^{(0)}$, and then mapped back by $A \rightarrow E$ mapping to a point in E-space, $e_p^{(1)}$. This point must fall within an E_p region because all points in A-space are possible and must be mapped to a point representing a possible state. This point may no longer on the E_c plane, and so the goal constraint is imposed again by projecting onto the E_c plane. This corresponds to clamping of some E-units to the goal. Using this point as the new starting point, the process is repeated. If it finds a solution after some iteration, the point no longer moves because a point in the E_p regions is mapped to itself. From the assumption that the mappings generalize based on similarity between patterns, or in this case similarities defined as the distance between points, it can be shown that, the distance between the points on the E_c plane ($e_c^{(0)}$, $e_c^{(1)}$, $e_c^{(2)}$, $\cdots$) and the points in the E_p regions ($e_p^{(1)}$, $e_p^{(2)}$, ..) will keep decreasing. Thus, the network will either find a solution or fall into a local minimum, depending on the shape of the E_p regions in relation to the E_c plane.

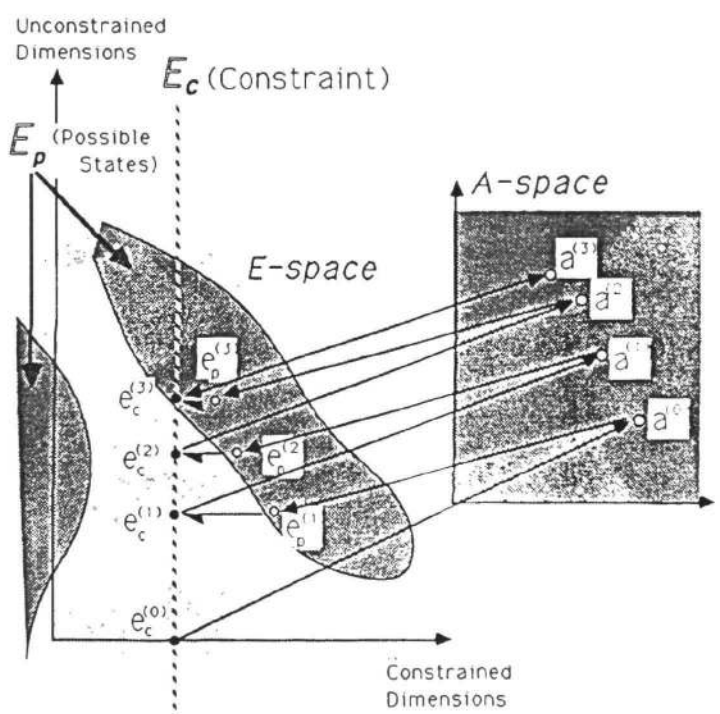

Figure 8. An analysis of the planning process. A solution is a point in E-space that both is possible (in an E_p region) and satisfies the goal (on the E_c plane). The network searches for such a point by mapping back and forth between E-space and A-space.

SUMMARY AND EXTENSION

The EAE network gives one explanation of how learning and planning of actions can be accomplished, using only feedback from the environment and when the desired state of the environment is only partially specified. Obviously, in order to evaluate the framework it is necessary to test on many different tasks and also to compare the performance of the network more closely to that of humans. The advantage of the EAE framework is that it does not need any domain specific structure, except a design of representation of the environment and of the actions, and thus it is readily applicable and testable in other domains. Furthermore, the framework can be extended in several interesting ways. First, it is obvious that people can modify a planned sequence of actions based on feedback from the environment, especially when the action is slow. The system could use feedback from the environment during the execution of an action plan to adjust its prediction of future states and the action plan. This would allow accurate performance without perfect prediction and therefore without perfect learning of the mappings. Second, the framework provides a possible way to model automatization of a skill when actions and environmental contexts are sufficiently correlated (Shiffrin & Schneider, 1977). In such situations, the system could establish a mapping from the contexts to actions, which could speed up, or possibly eliminate the need for, the process of generating predictions and comparing with explicit repesentations of goals.

REFERENCES

Anzai, Y. (1984). Cognitive control of real-time event-driven systems. *Cognitive Science, 8*, 221-254.

Chauvin, Y. (1987). *Generalization as a function of the number of hidden units*. Unpublished manuscript.

Cottrell, G. W., Munro, P., & Zipser, D. (1987). Image compression by back propagation: an example of extensional programming. In N. E. Sharkey (Ed.), *Review of Cognitive Science I*. Norwood, NJ: Ablex.

Hinton, G. E., & Smolensky, P. (1984). *Parallel computation and the mass-spring model of motor control.* Report 123, Center for Human Information Processing, UC 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). Amherst, MA.

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

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.

Miyata, Y. (1988). *The learning and planning of actions.* PhD thesis, Psychology Department, UC San Diego. Tech. Rep. No. 8802, Institute for Cognitive Science, UC San Diego.

Rosenbaum, D. A., Hindorff, V., & Munro, M. (1987). Scheduling and programming of rapid finger sequences: tests and elaborations of the hierarchical editor model. *Journal of Experimental Psychology: Human Perception and Performance, 13*, 193-203.

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.

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.

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

Shiffrin, R. M., & Schneider, W. (1977). Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. *Psychological Review, 84*, 127-190.