Training Recurrent Neural Networks with Temporal Input Encodings *

C.W. Omlin [a,b], C.L. Giles [a,c], B.G. Horne [a], L.R. Leerink [d], T. Lin [a]
[a] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540
[b] CS Department, Rensselaer Polytechnic Institute, Troy, NY 12180
[c] UMIACS, University of Maryland, College Park, MD 20742
[d] EE Department, The University of Sydney, Sydney, NSW 2006

***Abstract*— We investigate the learning of deterministic finite-state automata (DFA's) with recurrent networks with a single input neuron, where each input symbol is represented as a temporal pattern and strings as sequences of temporal patterns. We empirically demonstrate that obvious temporal encodings can make learning very difficult or even impossible. Based on preliminary results, we formulate some hypotheses about 'good' temporal encoding, i.e. encodings which do not significantly increase training time compared to training of networks with multiple input neurons.**

I. INTRODUCTION

Recurrent neural networks (RNN's) can be trained to behave like deterministic finite-state automata (DFA's) [2, 3, 4, 6, 8, 9, 10]. The symbols of the input alphabet have to be encoded into the network's set of input neurons. We investigate whether it is feasible to learn DFA's with a temporal encoding scheme where the example strings are fed into a network via a single input neuron. All input symbols are mapped into a sequence of temporal signals ('pulse-encoding'). This encoding scheme is biologically motivated, but it also represents an interesting machine learning problem in itself.

We empirically investigate different encodings and demonstrate what the limitations are for training RNN's with temporal encodings. Based on our empirical results, we formulate hypotheses which we suspect to be necessary conditions for successful learning. We perform the simulations only for second-order networks, but we believe that the results also apply to first-order networks.

* Proceedings of the 1994 IEEE International Conference on Neural Networks, p. 1267-1278. Copyright IEEE Press.

II. FINITE STATE AUTOMATA

A deterministic finite-state automaton (DFA) M is an acceptor of a regular language $L(M)$. Formally, a DFA M is a 5-tuple $M =< \Sigma, Q, R, F, \delta >$ where $\Sigma = \{a_1, \ldots, a_m\}$ is the alphabet of the language L, $Q = \{q_1, \ldots, q_n\}$ is a set of states, $R \,\epsilon\, Q$ is the start state, $F \subseteq Q$ is a set of accepting states and $\delta : Q \times \Sigma \rightarrow Q$ defines state transitions in M. A string x is accepted by the DFA M and hence is a member of the regular language $L(M)$ if an accepting state is reached after the string x has been read by M. For more details see [5].

III. TRAINING RECURRENT NETWORKS

A. Recurrent Network

We train discrete-time, recurrent networks with second-order weights W_{ijk} to behave like DFA's [2, 3, 4, 6, 8, 9, 10]. A network accepts a time-ordered sequence of inputs and evolves with dynamics defined by the following equations:

$$S_i^{(t+1)} = h(a_i(t)) = \frac{1}{1 + e^{-a_i(t)}} \qquad (1)$$

$$a_i(t) = b_i + \sum_{j,k} W_{ijk} S_j^{(t)} I_k^{(t)}, \qquad (2)$$

where b_i is the bias associated with hidden recurrent state neurons S_i and I_k is an input neuron. A special state neuron S_0 is the designated output neuron. Given a set of positive and negative example strings, the network is trained using the backpropagation-through-time algorithm (BPTT) [7]. Training proceeds until the network correctly classifies all example strings within a certain tolerance ε; typically, $\varepsilon = 0.1$. After the training phase, we relax the tolerance of the values of the output neurons; a trained network accepts a string if the value of S_0 at the end of the string is greater than 0.5; otherwise, the network rejects the string.

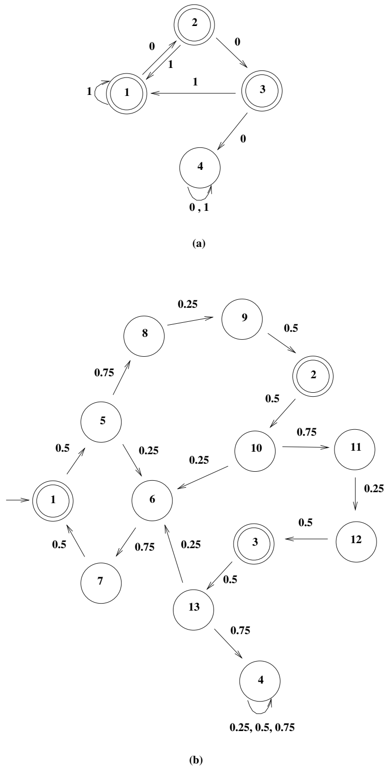

Figure 1: **Transformation of DFA's:** The DFA M shown in figure a accepts all strings over the alphabet $\{0,1\}$ which do not contain '000' as a substring. State 1 is the start state. Accepting strings are shown with double circles (states 1, 2, 3); state 4 is the only rejecting state. The DFA M was transformed into the minimized DFA M^* shown in figure b under the input encoding $0 \rightarrow \{0.5, 0.75, 0.25, 0.5\}$ and $1 \rightarrow \{0.5, 0.25, 0.75, 0.5\}$.

IV. INPUT ENCODINGS

A. Spatial Encoding

Under a spatial encoding of the input symbols, the alphabet of the DFA is mapped into a set of input neurons: $\Sigma \rightarrow \{I_1, I_2, \ldots I_K\}$. A one-to-one mapping $a_k \rightarrow I_k$ is often used, where only one input neuron I_k corresponding to input symbol a_k has a high signal at any given time step. This orthonormal input symbol encoding is an extreme case of spatial encoding. As each symbol is presented to the network, a different network computes the new network state. The operation $S_j^{(t)} I_k^{(t)}$ directly corresponds to a DFA state transition $\delta(q_j, a_k) = q_i$.

B. Temporal Encoding

We encode symbols of a DFA as sequences of temporal signals; a representative example is shown in figure 3. This encoding of input symbols transforms the original DFA M into a DFA M^* with more states and more transitions. Formally, a DFA $M =< \Sigma, Q, R, F, \delta >$ is transformed into a DFA $M^* =< \Pi, Q \cup X, R, F, \gamma >$. The new input alphabet $\Pi = \{b_1, b_2, \ldots, b_T\}$ is the set of all possible values of the temporal signal; it is understood that each input symbol has a distinct encoding $a_k \rightarrow b_k^1 b_k^2 \ldots b_k^P$, i.e. i.e. no two input encodings are the same for any pair of input symbols a_k and a_l. We will refer to b_k^p as the p^{th} component of the temporal encoding of symbol a_k. In general, different symbols may have pulse encodings of different length; for the remainder of this paper, we assume that the pulse encodings of all input symbols of M have the same length P. $Q \cup X$ is the extended set of states and γ is the new transition function. A state transition $\delta(q_j, a_k) = q_i$ in M is transformed into a new state transition $\gamma(\ldots(q_j, b_k^1), b_k^2, \ldots), b_k^P) = q_i$ in M^*. We will assume that no illegal encodings will occur, i.e. we do not need to specify state transitions $\gamma(q_l, b_k^p)$ if q_l is not a state in M or b_k^p is not the p^{th} component of the encoding of some symbol a_k.

The DFA M shown in figure 1a accepts all strings over the alphabet $\{0,1\}$ which do not contain '000' as a substring. Under a temporal input encoding, DFA M is transformed into the minimized DFA M^* shown in figure 1b. A temporal encoding of input symbols generally causes an increase in the number of states and transitions in the transformed DFA.

V. SIMULATIONS

We trained second-order RNN's with 20 recurrent neurons with one input neuron on the first 1,000 example strings in alphabetical order; the labels of the strings were assigned by the DFA shown in figure 1a. The weights were initialized with random values in the interval $[-0.1, 0.1]$. We experimented with different encodings of different lengths. We report here the results for 10 different training runs for each of the encoding schemes. We allowed a maximum of 5,000 epochs for each run; if a network failed to converge and the mean-squared error did not change until 5,000 epochs were reached, we concluded that the network was stuck in a local minimum and that further training would not lead to a solution.

Assuming the amplitude of all signals is limited to the interval $[0,1]$, each component b_k^p of the encoding of a symbol a_k has to be greater than zero. A signal $b_k^p = 0$ forces the state neurons to assume the constant value $S_i^{t+1} = h(-b_i)$ regardless of past values S_i^t; this makes previous DFA states indistinguishable and RNN's fail to converge. Notice that this is a property of second-order networks only; first-order networks do not share the restriction $b_k^p \neq 0$. We will discuss in the following sections different encod-

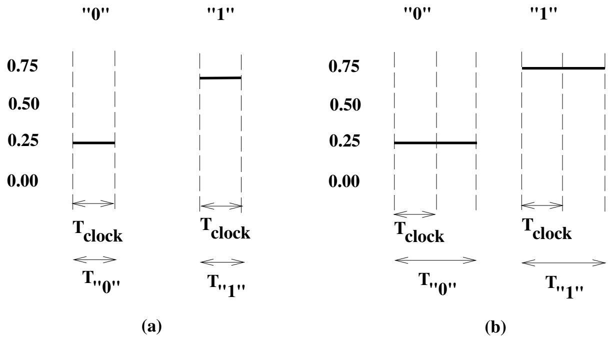

Figure 2: **Uniquely Identifiable Encodings:** These simple encodings allow the current symbol to be uniquely identified from the current temporal signal.

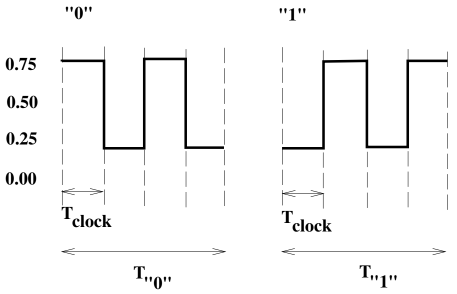

Figure 3: **Maximally Non-Identifiable Encoding:** This encoding does not allow the current symbol to be uniquely identified from the current temporal signal. The two encodings for '0' and '1' are out of phase, but otherwise they are identical.

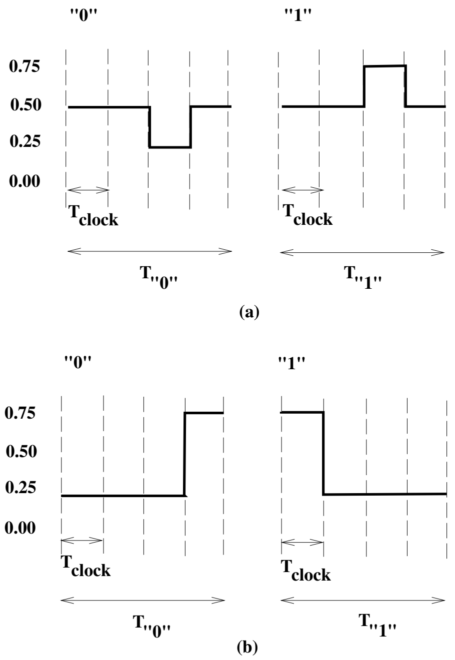

Figure 4: **Redundant Encodings:** These encodings contain redundant information which does not contribute to the distinction between input symbols '0' and '1'.

ing schemes and compare the convergence times for identical initial conditions. The number of possible encodings grows exponentially with the length of the symbol encodings; it is thus impossible to perform an exhaustive search for the best encoding. We have selected a few encodings which we think are interesting.

A. Symbol Identification

We call input symbol a_k identifiable if the symbol can uniquely be identified from any component b_k^p of the symbol's temporal encoding. The simplest such encoding for a DFA with input alphabet $\{0, 1\}$ is the encoding shown in figure 2a. The first row in table 1 shows that only 2 out of 10 RNN's converged with this simple encoding. We extended the length of each encoding from one to two time steps (figures 2b). We observed that convergence is not very likely with longer signal encoding either. Similarly, we found that convergence for the symbol encoding shown in figure 3 is difficult. The two symbol encodings are out of phase, but otherwise they are identical. A symbol a_k cannot be identified from a signal b_k^p alone. We observed that only 3 of the 10 runs converged to a solution.

B. Signal Redundancy

We encoded input symbols according to figure 4a. The information which distinguishes the encodings for the two symbols from each other is the same as in figure 2a. The remainder of the encodings is redundant information which does not contribute to distinguishing '0' from '1'; none of the 10 runs converged to a solution.

C. Signal Variation

We now consider symbol encodings where no two adjacent signals b_k^p and b_k^{p+1} are the same within the encoding of a_k. Two representative examples are shown in figures 5a and 5b. 8 and 6 out of 10 runs converged for the encodings of figures 5a and 5b, respectively. In runs where both encodings converged, the encoding 5a found a solution faster.

D. Signal Length

We shortened encodings of figures 4, 5a, and 5b by removing the first and the last components of the encodings. Compared to the longer encodings, encodings shown in figures 6a and 6b are more likely to converge and the RNN's learn faster. None of the 10 RNN's converged for the encoding shown in figure 6c.

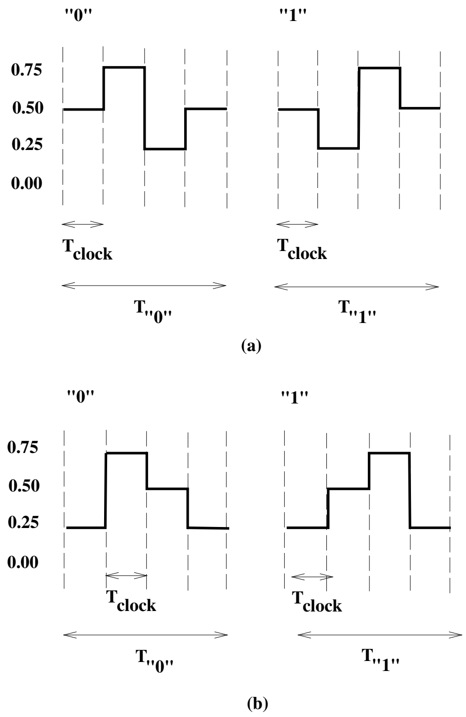

Figure 5: **Signal Variation**: No two adjacent signals of a symbol encoding are the same.

E. Spatial Distribution of Input Symbols

For comparison, we trained 10 RNN's with 20 recurrent state neurons; the inputs were spatially encoded with 2 input neurons, one for each of the input symbols. The last row of table 1 shows the convergence times for 10 different networks; their initial conditions were not the same as in the previous experiments, since we changed the network architecture.

F. Discussion

The above experiments clearly show that BPTT prefers some input symbol encodings over others. We would like to be able to characterize the encodings which generally yield fast convergence. We attempted to analyze the symbol encodings discussed above with standard techniques. Neither a linear correlation analysis nor a Discrete Fourier Transformation have revealed any significant properties of 'good' encodings.

In the following, we say a RNN has learned a DFA if the training converged on a given set of strings; we are not concerned with the generalization performance of trained RNN's. Based on our empirical evidence, we will now formulate justified hypotheses about 'good' temporal encodings of input signals, i.e. encodings which do not significantly increase the learning time compared to learning a data set under an orthonormal spatial input symbol encoding. While a spatial encoding will always increase the number of weights, we will assume that network

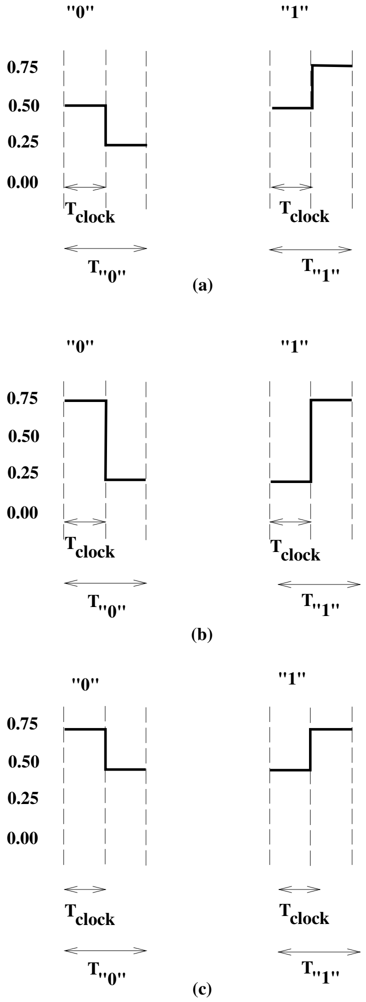

Figure 6: **Encodings without Redundancy**: The above encodings correspond to the encodings in figures 4a, 5a and 5b without the first and last component.

sizes for spatial and temporal symbol encodings are comparable.

Hypothesis V.1 *The larger a DFA, the more difficult it is to learn it with a RNN. The opposite is not necessarily true for temporal encodings of input symbols.*

From experience, we know that learning larger DFA's requires larger RNN in order for the network to build an internal representation of the DFA states. We have found a temporal encoding which does not change the size of the DFA, yet the RNN's had difficulty learning the DFA.

Hypothesis V.2 *A RNN can learn a DFA more easily if the input symbol encoding is either spatially or temporally distributed.*

Usually, we encode input symbols discretely over a set of input neurons. A one-to-one mapping is an extreme case of spatial distribution of input symbols. Under that encoding, a RNN is likely to learn small DFA's (last row of table 1). We have an example of an symbol encoding which was neither spatially nor temporally distributed; the RNN's had difficulty to converge.

Hypothesis V.3 *Unique identification of symbols from the current signal is neither a necessary nor a sufficient condition for likely convergence of RNN's.*

RNN's where the symbols were easily detectable from the current signal of the temporal encoding had difficulty to converge. On the other hand, we successfully trained RNN's where the symbol was not uniquely identifiable from the current signal.

Hypothesis V.4 *A temporal input encodings with redundant information makes RNN's less likely to converge.*

Theoretical results on learning sequences with RNN's suggest that learning algorithms based on gradient-descent have difficulties propagating error information over long signal spans, making it hard to distinguish DFA states [1].

Hypotheses V.1 and V.3 may seem surprising. Intuitively, we expect RNN's to converge easily on the simplest possible temporal encoding; furthermore, one could assume that a network prefers an encoding which allow the symbol to be uniquely identified from a single component of its encoding.

There exists a trade-off between temporal distribution of input symbol encodings (hypothesis V.2) and the need for encoding without redundancy (hypothesis V.4). We hypothesize that a distribution of input signals - spatial or temporal - is a necessary condition for likely learning of DFA's with RNN's. However, a distribution which is spread too widely, will hinder gradient-based learning algorithms.

The ultimate goal is to identify properties of temporal input symbol encodings which make convergence no more difficult than spatial input symbol encodings.

VI. SIGNAL DELAYS

It is an interesting question whether a network can learn a DFA if random delays between signals are introduced. In addition to learning to behave like a DFA, a network has to learn to ignore signal delays.

For the experiments, we augmented the signal alphabet Π by a special signal b_{delay} which separates consecutive, complete temporal encodings of input symbols. We encoded the training strings according to the encoding shown in figure 5a and randomly separated the complete encodings of each input symbol by at most one time step using $b_{delay} = 0.5$. In general, one would allow random signal delays of arbitrary length.

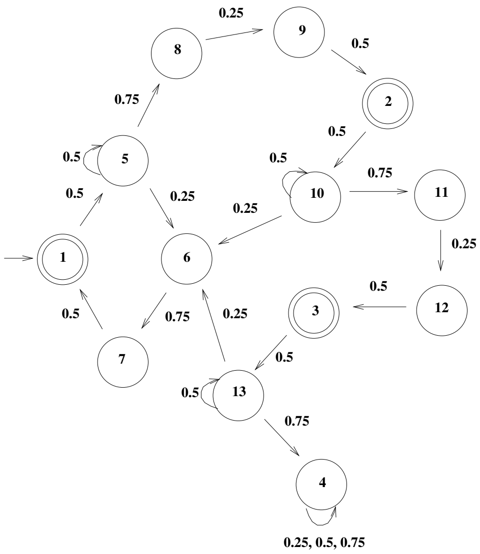

Figure 7: **DFA Allowing Random Signal Delays:** The shown DFA M allows signal delays of random length between complete input symbol encodings. The delay encoding is $b_{delay} = 0.5$. The DFA differs from the DFA shown in figure 1b only in the transitions from states 5, 10 and 13 on input signal $b_{delay} = 0.5$.

There are two problems associated with random signal delays: First, the delays always add to the length of each example string making learning with gradient-descent more difficult. Second, the DFA's to be learned with and without signal delays are not identical. Random signal delays always add new state transitions as follows: Let q_i be any state reached on the last signal b_k^P of the encoding of symbol a_k; a new state transition $\gamma(q_i, b_{delay}) = q_i$ is added. The DFA in figure 1b is extended to allow delays of arbitrary length between complete encodings of input symbols (figure 7).

We found that none of the RNN's converged when at the most one signal delay was inserted between two consecutive symbols in the training data. The slight increase in overall DFA complexity (number of states and transitions) alone can hardly be responsible for the RNN's failure to converge. The increased length of the training strings due the introduction of signal

encoding	run 1	run 2	run 3	run 4	run 5	run 6	run 7	run 8	run 9	run 10
temporal 2a	-	-	-	-	-	340	-	856	-	-
temporal 2b	-	-	-	-	459	-	-	-	-	-
temporal 3	-	-	-	-	-	344	1280	3456	-	-
temporal 4a	-	-	-	-	-	-	-	-	-	-
temporal 4b	-	-	-	-	-	1011	-	1382	-	-
temporal 5a	241	221	185	-	844	325	372	-	247	1505
temporal 5b	-	565	999	723	-	2942	615	-	1322	-
temporal 6a	326	305	287	315	-	236	265	296	315	358
temporal 6b	117	116	140	144	113	119	107	135	121	116
temporal 6c	-	-	-	-	-	-	-	-	-	-
signal delay (5a)	-	-	-	-	-	-	-	-	-	-
spatial	54	45	31	42	35	47	46	36	47	36

Table 1: **Input Symbol Encodings**: The table shows the convergence of 10 RNN's under different input symbol encodings. The same initial networks were used for training with different encodings. The numbers in the first column refer to the number of the figure that describes the encoding of input symbols '0' and '1'.

delays could explain the RNN's difficulty to converge. However, the training strings with signal delays are still shorter than than the ones used for symbol encodings of length 4 for some of which convergence was likely. Thus, we conclude that neither the increase in DFA complexity nor the increased string length alone can be responsible for the RNN's failure to converge; other characteristics of the input symbol encoding must play a role in determining whether a network is likely to converge.

VII. CONCLUSIONS

We have started the investigation of training recurrent neural networks (RNN's) with temporally encoded input sequences. Our preliminary empirical results have shown some surprising results: RNN's could not easily learn a simple deterministic finite-state automaton (DFA) with the shortest possible temporal encoding of the input sequences although similar RNN's with a spatial input symbol encoding learned easily. We speculate that there exist properties of temporal input signal encodings which make RNN's more or less likely to converge. One such hypothesis may be that input symbols must be distributed - ether spatially or temporally - in order for RNN's to be likely to learn a DFA. Based on that assumption, we further speculate that there exists a trade-off between convergence and a wide distribution which leads to signal redundancy. There exist theoretical result which suggest that training with gradient-descent algorithms becomes harder with increasing sequence length. Thus, short input sequence encodings should be favored. Standard techniques for analyzing signals (correlation analysis and Discrete Fourier Transformation) have not revealed any characteristics of encodings which allow fast learning; further research is needed to find out why a network prefers certain encodings of input signals over others.

VIII. ACKNOWLEDGMENT

We would like to thank J. L. Johnson for suggesting the investigation of training recurrent networks with temporal input encodings.

References

[1] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *IEEE International Conference on Neural Networks*, vol. III, (Piscataway, NJ), pp. 1183–1188, IEEE Press, 1993.

[2] A. Cleeremans, D. Servan-Schreiber, and J. McClelland, "Finite state automata and simple recurrent recurrent networks," *Neural Computation*, vol. 1, no. 3, pp. 372–381, 1989.

[3] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[4] C. Giles, C. Miller, D. Chen, H. Chen, G. Sun, and Y. Lee, "Learning and extracting finite state automata with second-order recurrent neural networks," *Neural Computation*, vol. 4, no. 3, p. 380, 1992.

[5] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley Publishing Company, Inc., 1979.

[6] J. Pollack, "The induction of dynamical recognizers," *Machine Learning*, vol. 7, pp. 227–252, 1991.

[7] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, ch. 8, Cambridge, MA: MIT Press, 1986.

[8] D. Servan-Schreiber, A. Cleeremans, and J. McClelland, "Graded state machine: The representation of temporal contingencies in simple recurrent networks," *Machine Learning*, vol. 7, p. 161, 1991.

[9] R. Watrous and G. Kuhn, "Induction of finite-state languages using second-order recurrent networks," *Neural Computation*, vol. 4, no. 3, p. 406, 1992.

[10] Z. Zeng, R. Goodman, and P. Smyth, "Learning finite state machines with self-clustering recurrent networks," *Neural Computation*, vol. 5, no. 6, pp. 976–990, 1993.