Extracting Finite State Representations from Recurrent Neural Networks trained on Chaotic Symbolic Sequences

Peter Tiňo* **Miroslav Köteles**
Department of Computer Science and Engineering
Slovak Technical University
Ilkovicova 3, 812 19 Bratislava, Slovakia
Email: `tino,koteles@dcs.elf.elf.stuba.sk`

To appear in IEEE Transactions on Neural Networks, 1999.

Abstract

While much work has been done in neural based modeling of real valued chaotic time series, little effort has been devoted to address similar problems in the symbolic domain. We investigate the knowledge induction process associated with training recurrent neural networks (RNNs) on single long chaotic symbolic sequences. Even though training RNNs to predict the next symbol leaves the standard performance measures such as the mean square error on the network output virtually unchanged, the networks nevertheless do extract a lot of knowledge. We monitor the knowledge extraction process by considering the networks stochastic sources and letting them generate sequences which are then confronted with the training sequence via information theoretic entropy and cross-entropy measures. We also study the possibility of reformulating the knowledge gained by RNNs in a compact and easy-to-analyze form of finite state stochastic machines. The experiments are performed on two sequences with different "complexities" measured by the size and state transition structure of the induced Crutchfield's ϵ-machines. We find that, with respect to the original RNNs, the extracted machines can achieve comparable or even better entropy and cross-entropy performance. Moreover, RNNs reflect the training sequence complexity in their dynamical state representations that can in turn be reformulated using finite state means. Our findings are confirmed by a much more detailed analysis of model generated sequences through the statistical mechanical metaphor of entropy spectra. We also introduce a visual representation of allowed block structure in the studied sequences that, besides having nice theoretical properties, allows on the topological level for an illustrative insight into both RNN training and finite state stochastic machine extraction processes.

1 Introduction

The search for a model of an experimental time series is a problem that has been studied for some time. At the beginning, the linear models were applied [5], followed later by nonlinear ones [42], in particular, neural networks [32].

*currently with *Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Vienna, Austria*

Often the modeling task is approached by training a nonlinear learning system with a predictive mapping. When dealing with chaotic time series, one runs into trouble of not having a capable theory to guarantee if the predictor has successfully identified the original model [32]. The simple point by point comparison between the original and predicted time series can no longer be used as a goodness of fit measure [33]. The reason is simple: two trajectories produced by the same dynamical system may be very different point-wise and still evolve around the same chaotic attractor. In this case, Principe and Kuo [32] suggest to take a more pragmatic approach. Seed the predictor with a point in state space, feed the output back to its input and generate a new time series. To compare the new and original (training) time-series, use the dynamic invariants, such as correlation dimension and Lyapunov exponents. Dynamic invariants measure *global* properties of the attractor and are appropriate tools in quantifying the success of dynamic *modeling*.

The main difference between prediction of time series and dynamic modeling is that while the former is primarily concerned with the instantaneous prediction error, the latter is interested in *long term* behavior.

Much work has been done in neural modeling of real valued chaotic time series [40], but little effort has been devoted to address similar problems in the symbolic domain: given a long sequence S with positive entropy, i.e. difficult to predict, construct a stochastic model that has the information theoretic properties similar to those of S. So far, connectionist approaches to extraction of useful statistics from long symbolic sequences have have been primarily concerned with compression issues [37] or detection and analysis of significant subsequences [20, 38].

In this paper, we study three main issues associated with training RNNs on *long* chaotic sequences.

1. Chaotic sequences are by nature unpredictable. Consequently, one can hardly expect RNNs to be able to *exactly* learn the prediction task[1] on such sequences. On the other hand, even though training the networks on long chaotic sequences to predict[2] the next symbol leaves the traditional performance measures such as the mean square error or the mean absolute error virtually unchanged, RNNs nevertheless do learn important structural properties of the training sequence and are able to generate sequences mimicking the training sequence in the information theoretic sense. Therefore, we monitor the RNN training process by comparing the entropies of the training and network generated sequences and by computing their statistical "distances" through the cross-entropy estimates.

2. It is a common practice among recurrent network researchers applying their models in grammatical inference tasks to extract finite state representations of trained networks by identifying clusters in recurrent neurons' activation space (RNN state space) with states of the extracted machines [31]. This methodology was analyzed by Casey [7] and Tiňo et al. [28]. Casey shows that in general such a heuristic may not work. He also proves[3] that whenever a finite state automaton defining a regular language $\mathcal{L}$ contains a loop, there must be an associated attractive set inside the state space of a RNN recognizing $\mathcal{L}$. Furthermore, it is shown that the method introduced in [15] for extracting an automaton from trained RNNs is consistent. The method is based on dividing RNN state space into equal hypercubes and there is always a *finite* number of hypercubes one needs to unambiguously cover the regions of equivalent states. Tiňo et al.

[1] Tani and Fukumara [39] trained recurrent networks on a set of *short* pieces taken from a binary sequence generated by the logistic map via the critical point defined generating partition. Short length of the symbolic extracts enabled the authors to train the networks to become *perfect* next symbol predictors on the training data. The inherently unpredictable nature of the chaotic mother sequence was reflected by the induced chaotic dynamics in the recurrent network state space.

[2] given the current symbol

[3] for RNNs operating in a noisy environment representing for example a noise corresponding to round-off errors in computations performed on a digital computer

attempt to explain the mechanism behind the often reported performance enhancement when using extracted finite state representations instead of their RNN originals. Loosely speaking, for small time scales, saddle-type sets can take on the roles of attractive sets and lead, for example, to introduction of loops in the state transition diagrams of the extracted machines, just as the attractive sets do. On longer time scales, the extracted machines do not suffer from stability problems caused by the saddle-type dynamics. We test whether these issues, known from the RNN grammatical inference experiments, translate to the problem of training RNNs on single long chaotic sequences. To this end, we extract from trained RNNs their finite state representations of increasing size and compare the RNN and extracted machine generated sequences through information theoretic measures.

The experiments are performed on two chaotic sequences with different levels of computational structure expressed in the induced ϵ-machines. The ϵ-machines were introduced by Crutchfield et al. [11, 12] as computational models of dynamical systems. Size of the extracted machines from RNN that attain comparable performance to that of the original RNN serves as an indicator of the "neural finite state complexity" of the training sequence. We study the relationship between the computational ϵ-machine and RNN-based notions of dynamical system complexities.

3. We compare predictive models by confronting the model generated sequences with the training sequence. As already mentioned, we use the level of sequence unpredictability (expressed through the entropy) and the statistical "distance" (expressed via cross-entropy) between the model generated sequences and the training sequence as the first indicators of the model performance. To get a clearer picture of the subsequence distribution structure, we extend the sequence analysis in two directions motivated by the statistical mechanics and the (multi)fractal theory.

   (a) A statistical mechanical metaphor of *entropy spectra* [43, 4] enables us to study the block structure in long symbolic sequences across various block lengths and probability levels.

   (b) We visualize the structure of allowed finite length blocks through the so called *chaos block representations*. These representations are sequences of points in a unit hypercube corresponding in a one-to-one manner to the set of allowed blocks such that points associated with blocks sharing the same suffix lie in a close neighborhood. The chaos block representations are used to track down the evolving ability of RNNs to confine their dynamics in correspondence with the set of allowed blocks in the training sequence and to visualize the effect of RNN finite state reformulations.

The plan of this paper is as follows. We begin in section 2 with a brief introduction to entropy measures on symbolic sequences and continue in section 3 with description of the RNN architecture used in this study. The algorithm for extracting finite state stochastic machines from trained RNNs is presented in section 4. Section 5 covers the experiments on two chaotic sequences. In section 6 we introduce a visual representation of allowed blocks in the studied sequences and apply the methodology to experimental data from section 5. Conclusion in section 7 summarizes the key insights in this study.

2 Statistics on symbolic sequences

We consider sequences $S = s_1 s_2 ...$ over a finite alphabet $\mathcal{A} = \{1, 2, ..., A\}$ (i.e. every symbol s_i is from $\mathcal{A}$) generated by stationary information sources [18]. The sets of all sequences over $\mathcal{A}$ with a finite number of symbols and exactly n symbols are denoted by $\mathcal{A}^+$ and $\mathcal{A}^n$, respectively. By S_i^j, $i \leq j$, we denote the string $s_i s_{i+1} ... s_j$, with $S_i^i = s_i$.

Denote the (empirical) probability of finding an n-block $w \in \mathcal{A}^n$ in S by $P_n(w)$. A string $w \in \mathcal{A}^n$ is said to be an allowed n-block in the sequence S, if $P_n(w) > 0$. The set of all allowed n-blocks in S is denoted by $[S]_n$.

Statistical n-block structure in a sequence S is usually described through generalized entropy spectra. The spectra are constructed using a formal parameter β that can be thought of as the inverse temperature in the statistical mechanics of spin systems [12].

The original distribution of n-blocks, $P_n(w)$, is transformed to the "twisted" distribution [43] (also known as the "escort" distribution [4])

$$Q_{\beta,n}(w) = \frac{P_n^\beta(w)}{\sum_{v \in [S]_n} P_n^\beta(v)}. \tag{1}$$

The entropy rate[4]

$$h_{\beta,n} = \frac{-\sum_{w \in [S]_n} Q_{\beta,n}(w) \log Q_{\beta,n}(w)}{n} \tag{2}$$

of the twisted distribution $Q_{\beta,n}$ approximates the thermodynamic entropy density[5] [43]

$$h_\beta = \lim_{n \to \infty} h_{\beta,n}. \tag{3}$$

When $\beta = 1$ (metric case), $Q_{1,n}(w) = P_n(w)$, $w \in \mathcal{A}^n$, and h_1 becomes the metric entropy of subsequence distribution in the sequence S. The infinite temperature regime ($\beta = 0$), also known as the topological, or counting case, is characterized by the distribution $Q_{0,n}(w)$ assigning equal probabilities to all allowed n-blocks. The topological entropy h_0 gives the asymptotic growth rate of the number of distinct n-blocks in S as $n \to \infty$.

Varying the parameter β amounts to scanning the original n-block distribution P_n – the most probable and the least probable n-blocks become dominant in the positive zero ($\beta = \infty$) and the negative zero ($\beta = -\infty$) temperature regimes, respectively. Varying β from 0 to ∞ amounts to a shift from all allowed n-blocks to the most probable ones by accentuating still more and more probable subsequences. Varying β from 0 to $-\infty$ accentuates less and less probable n-blocks with the extreme of the least probable ones.

For each value of β, the hight of the entropy spectrum $h_{\beta,n}$ of a sequence S corresponds to the average symbol uncertainty in randomly chosen n-blocks from S at the temperature $1/\beta$. The steepness of the spectrum, as $\beta > 0$ grows, tells us how rich is the probabilistic structure of high-probability n-blocks in S. If there are many almost equally probable high probability n-blocks, then the entropy spectrum will decrease only gradually and the less structured are the n-block probabilities, the slower is the decrease in $h_{\beta,n}$. On the other hand, a rich probability structure in high probability n-blocks with many different probability levels and few dominant most frequent n-blocks will be reflected by a steeply decreasing entropy spectrum.

The same line of thought applies to the negative part of the entropy spectrum ($\beta < 0$). In this case, the decreasing rate of the spectrum (as $|\beta|$ grows) reflects the richness of the probability structure in low-probability n-blocks of the sequence S.

2.1 Block length independent Lempel-Ziv entropy and cross entropy estimates

For a stationary ergodic process that has generated a sequence $S = s_1 s_2 ... s_N$ over a finite alphabet $\mathcal{A}$, the Lempel-Ziv codeword length for S, divided by N, is a computationally efficient and reliable

[4] in this study we measure the information in bits, hence $\log \equiv \log_2$

[5] thermodynamic entropy density is directly related to the β-order Rényi entropy [34] (cf. [43])

estimate of the source entropy (h_1, eq. (3)) [44]. In particular, let $c(S)$ denote the number of phrases in S resulting from the incremental parsing of S, i.e. sequential parsing of S into distinct phrases such that each phrase is the shortest string which is not a previously parsed phrase. For example, let $S = 12211221211$. Then, the parsing procedure yields phrases $1, 2, 21, 12, 212, 11$ and $c(S) = 6$. The Lempel-Ziv codeword length for S is approximated with $c(S) \log c(S)$ [44]. Hence, the Lempel-Ziv approximation of the source entropy is then

$$h_{LZ}(S) = \frac{c(S) \log c(S)}{N}. \tag{4}$$

The notion of "distance" between distributions used in this paper is a well-known measure in information theory, called Kullback-Leibler divergence. It is also known as the relative, or cross-entropy. Let P and Q be two Markov probability measures, each of some (unknown) finite order. The divergence between P and Q is defined by

$$d^{KL}(Q|P) = \limsup_{n \to \infty} \frac{1}{n} \sum_{w \in \mathcal{A}^n} Q_n(w) \log \frac{Q_n(w)}{P_n(w)}. \tag{5}$$

d^{KL} measures the expected additional code length required when using the ideal code for P instead of the ideal code for the "right" distribution Q.

Suppose we have only length-N realizations S_P and S_Q of P and Q respectively. Analogically to Lempel-Ziv entropy estimation (4), there is an estimation procedure for determining $d^{KL}(Q|P)$ from S_P and S_Q [44]. The procedure is based on Lempel-Ziv sequential parsing of S_Q with respect to S_P. First, find the longest prefix of S_Q that appears in S_P, i.e. the largest integer m such that the m-blocks $(S_Q)_1^m$ and $(S_P)_i^{i+m-1}$ are equal, for some i. $(S_Q)_1^m$ is the first phrase of S_Q with respect to S_P. Next, start from the position $m+1$ in S_Q and find, in a similar manner, the longest prefix $(S_Q)_{m+1}^k$ that appears in S_P, and so on. The procedure terminates when S_Q is completely parsed with respect to S_P. Let $c(S_Q|S_P)$ denote the number of phrases in S_Q with respect to S_P. Then, the Lempel-Ziv estimate of $d^{KL}(Q|P)$ is computed as [44]

$$d_{LZ}^{KL}(S_Q|S_P) = \frac{c(S_Q|S_P) \log N}{N} - h_{LZ}(S_Q). \tag{6}$$

Ziv and Merhav [44] proved that if S_Q and S_P are independent realizations of two finite order Markov processes Q and P, $d_{LZ}^{KL}(S_Q|S_P)$ converges (as $N \to \infty$) to $d^{KL}(Q|P)$ almost surely.

Although, in general, the processes in our experiments were not Markov of finite order, the Lempel-Ziv cross entropy estimate d_{LZ}^{KL} turned out to be a reasonably good indicator of statistical distance between model sequences.

3 Recurrent neural network

The recurrent neural network (RNN) presented in figure 1 was shown to be able to learn mappings that can be described by finite state machines [29].

We use a unary encoding of symbols from the alphabet $\mathcal{A}$ with one input and one output neuron devoted to each symbol. The input and output vectors $I^{(t)} = (I_1^{(t)}, ..., I_A^{(t)}) \in \{0, 1\}^A$ and $O^{(t)} = (O_1^{(t)}, ..., O_A^{(t)}) \in (0, 1)^A$, correspond to the activations of A input and output neurons respectively. There are two types of hidden neurons in the network.

- B hidden non-recurrent second-order neurons H_1,...,H_B, activations of which are denoted by $H_j^{(t)}$, $j = 1, ..., B$.
- L hidden recurrent second-order neurons R_1,...,R_L, called state neurons. We refer to the activations of state neurons by $R_i^{(t)}$, $i = 1, ..., L$. The vector $R^{(t)} = (R_1^{(t)}, ..., R_L^{(t)})$ is called the state of the network.

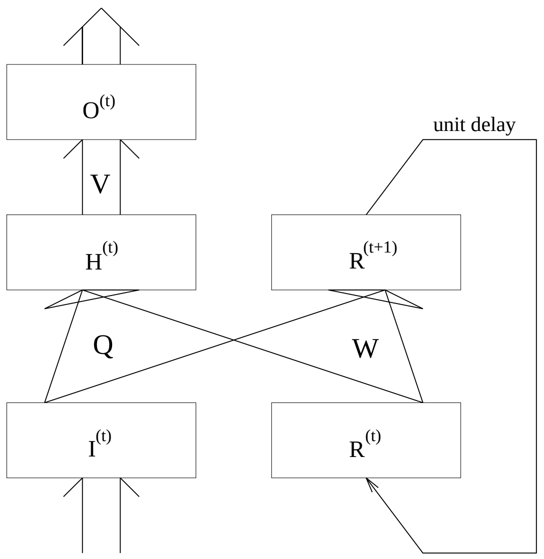

Figure 1: RNN architecture.

W_{iln}, Q_{jln} and V_{mk} are real-valued weights and g is a sigmoid function $g(x) = 1/(1+e^{-x})$. The activations of hidden non-recurrent neurons are determined by

$$H_j^{(t)} = g\left(\sum_{l,n} Q_{jln} R_l^{(t)} I_n^{(t)}\right). \tag{7}$$

The activations of state neurons at the next time step $(t+1)$ are computed as follows:

$$R_i^{(t+1)} = g\left(\sum_{l,n} W_{iln} R_l^{(t)} I_n^{(t)}\right). \tag{8}$$

The network output is determined by

$$O_m^{(t)} = g\left(\sum_{k} V_{mk} H_k^{(t)}\right). \tag{9}$$

The architecture strictly separates the input history coding part - *state network*, consisting of layers $I^{(t)}$, $R^{(t)}$, $R^{(t+1)}$ - from the part responsible for the association of the so far presented inputs[6] with the network output - *association network*, composed of layers $I^{(t)}$, $R^{(t)}$, $H^{(t)}$, $O^{(t)}$.

The network is trained on a single long symbolic sequence to predict, at each point in time, the next symbol. Training is performed via an on-line (RTRL-type [16]) optimization with respect to the error function

$$E(Q,W,V) = \frac{1}{2}\sum_m (T_m^{(t)} - O_m^{(t)})^2, \tag{10}$$

where $T_m^{(t)} \in \{0,1\}$ is the desired response value for the m-th output neuron at the time step t.

To start the training, the initial network state $R^{(1)}$ is randomly generated. The network is reset with $R^{(1)}$ at the beginning of each training epoch. During the epoch, the network is presented with the training sequence $S = s_1 s_2 s_3 \ldots$, one symbol per time step. Based on the current input[7] $I^{(t)}$ and state $R^{(t)}$, the network computes the output $O^{(t)}$ and the next state $R^{(t+1)}$. The desired response $T^{(t)}$ is the code of the next symbol s_{t+1} in the training sequence.

[6] the last input $I^{(t)}$, together with a code $R^{(t)}$ of the recent history of past inputs

[7] which is the code of the symbol s_t in S

After the training, the network is seeded with the initial state $R^{(1)}$ and the input vector coding the symbol s_1. For the next T_1 "pre-test" steps, only the state network is active: for the current input and state, the next network state is computed, and it comes into play, together with the next symbol from S, at the next time step. This way, the network is given a right "momentum" in the state path starting in the initial "reset" state $R^{(1)}$.

After T_1 pre-test steps, the network generates a symbol sequence by itself. In each of T_2 test steps, the network output is interpreted as a new symbol that will appear at the net input at the next time step. The network state sequence is generated as before. The output activations $O^{(t)} = (O_1^{(t)}, ..., O_A^{(t)})$ are transformed into "probabilities" $P_i^{(t)}$,

$$P_i^{(t)} = \frac{O_i^{(t)}}{\sum_{j=1}^{A} O_j^{(t)}}, \quad i = 1, 2, ..., A, \tag{11}$$

and the new symbol $\hat{s}_t \in \mathcal{A} = \{1, ..., A\}$ is generated with respect to the distribution $P_i^{(t)}$,

$$Prob\,(\hat{s}_t = i) = P_i^{(t)}, \; i = 1, 2, ..., A. \tag{12}$$

4 Extracting stochastic machines from RNNs

Transforming trained RNNs into finite-state automata has been considered by many [8, 13, 15, 23, 41, 7], usually in the context of training recurrent networks as acceptors of regular languages. After training, the activation space of RNN is partitioned (for example using a vector quantization tool) into a finite set of compartments, each representing a state of a finite-state acceptor. Typically, this way one obtains a more stable and easy-to-interpret representation of what has been learned by the network. For a discussion of recurrent network state clustering techniques see [29, 31].

Frasconi et. al [14] reported an algorithm for extracting finite automata from trained recurrent radial basis function networks. The algorithm was based on K-means [1], a procedure closely related to the dynamic cell structures (DCS) algorithm [6] used in this study. The aim of K-means clustering run on a set of RNN state activations $\{R^{(t)} |\; 1 \leq t \leq T_1 + T_2\}$ in the test mode is a minimization of the loss

$$E_{VQ} = \frac{1}{2} \sum_{t=1}^{T_1+T_2} d_E^2(R^{(t)}, C(R^{(t)})), \tag{13}$$

where $C(R^{(t)}) \in \{b_1, ..., b_K\}$ is the codebook vector to which the network state $R^{(t)}$ is allocated, and d_E is the Euclidean distance. The algorithm partitions the RNN state space $(0,1)^L$ into M regions $V_1, ..., V_M$, also known as the Voronoi compartments [2] of the codebook vectors $b_1, ..., b_M \in (0,1)^L$,

$$V_i = \{x \in (0,1)^L |\; d_E(x, b_i) = \min_j d_E(x, b_j)\}. \tag{14}$$

All points in V_i are allocated[8] to the codebook vector b_i.

The DCS algorithm attempts not only to minimize the loss (13), but also to preserve the input space topology in the codebook neighborhood structure. Unlike in the traditional self-organizing feature maps [19], the number of quantization centers (codebook vectors) and the codebook neighborhood structure in DCS is not fixed, but as the learning proceeds, the codebook is gradually grown and the corresponding codebook topology is adjusted to mimic to topology of the training data.

In this study, we consider trained RNNs stochastic sources the finite state representations of which take the form of stochastic machines. Stochastic machines (SMs) are much like non-deterministic

[8] ties as events of measure zero (points land on the border between the compartments) are broken according to index order

finite state machines except that the state transitions take place with probabilities prescribed by a distribution τ. To start the process, the machine chooses an initial state according to the "initial" distribution π, and then, at any given time step after that, the machine is in some state i, and at the next time step moves to another state j outputting some symbol s, with the transition probability $\tau(i,j,s)$. Formally, a stochastic machine M is quadruple $M = (Q, \mathcal{A}, \tau, \pi)$, where Q is a finite set of states, $\mathcal{A}$ is a finite alphabet, $\pi : Q \to [0,1]$ is a probability distribution over Q, and $\tau : Q \times Q \times \mathcal{A} \to [0,1]$ is a mapping for which

$$\forall i \in Q; \sum_{j \in Q, s \in \mathcal{A}} \tau(i,j,s) = 1. \tag{15}$$

Throughout this paper, for each machine, there is a unique initial state, i.e. $\exists i \in Q; \pi(i) = 1$ and $\forall j \in Q \setminus \{i\}; \pi(j) = 0$.

The stochastic machine $M_{RNN} = (Q, \mathcal{A}, \tau, \pi)$ is extracted from the RNN trained on a sequence $S = s_1 s_2 s_3 \ldots$ using the following algorithm:

1. Assume the RNN state space has been quantized by running a DCS on RNN states recorded during the RNN testing.

2. The initial state is a pair (s_1, i_1), where i_1 is the index of the codebook vector defining the Voronoi compartment $V(i_1)$ containing the network "reset" state $R^{(1)}$, i.e. $R^{(1)} \in V(i_1)$. Set $Q = \{(s_1, i_1)\}$.

3. For T_1 pre-test steps $1 < t \leq T_1 + 1$

   - $Q := Q \cup \{(s_t, i_t)\}$, where $R^{(t)} \in V(i_t)$
   - add the edge[9] $(s_{t-1}, i_{t-1}) \to^{s_t} (s_t, i_t)$ to the topological skeletal state-transition structure of M_{RNN}.

4. For T_2 test steps $T_1 + 1 < t \leq T_1 + T_2 + 1$

   - $Q := Q \cup \{(\hat{s}_t, i_t)\}$, where $R^{(t)} \in V(i_t)$ and $\hat{s}_t$ is the symbol generated at the RNN output (see eq. (12)).
   - add the edge[10] $(\hat{s}_{t-1}, i_{t-1}) \to^{\hat{s}_t} (\hat{s}_t, i_t)$ to the set of allowed state-transitions in M_{RNN}.

The probabilistic structure is added to the topological structure of M_{RNN} by counting, for all state pairs $(p,q) \in Q^2$ and each symbol $s \in \mathcal{A}$, the number $N(p,q,s)$ of times the edge $p \to^s q$ was invoked while performing steps 3 and 4. The state-transition probabilities are then computed as

$$\tau(p,q,s) = \frac{N(p,q,s)}{\sum_{r \in Q, a \in \mathcal{A}} N(p,r,a)}. \tag{16}$$

The philosophy of the extraction procedure is to let the RNN act as in the test mode, and interpret the activity of RNN, whose states have been partitioned into a finite set, as a stochastic machine M_{RNN}. When T_1 is equal to the length of the training set minus 1, and $T_2 = 0$, i.e. when RNN is driven with the training sequence, we denote the extracted machine by $M_{RNN(S)}$.

The machines M_{RNN} are finite state representations of RNNs working autonomously in the test regime. On the other hand, the machines $M_{RNN(S)}$ are finite state "snapshots" of the dynamics inside RNNs while processing the training sequence. In this case, the association part of RNN is used only during the training to induce the recurrent weights and does not come into play when extracting the machine $M_{RNN(S)}$ after the training process has finished.

[9] from the state (s_{t-1}, i_{t-1}) to the state (s_t, i_t), labeled with s_t

[10] $\hat{s}_{T_1+1} = s_{T_1+1}$

5 Experiments

5.1 Laser data

In the first experiment, we used Santa Fe competition laser data available on the Internet[11]. Since our primary concern is statistically faithful modeling, we dealt with the test continuation series of approximately 10,000 points. The competition data were recorded from a laser in a chaotic state. We converted the data into a time series of differences between the consecutive laser activations. The signal range, approximately $[-200, 200]$, was partitioned into 4 regions $[0, 50)$, $[50, 200)$, $[-64, 0)$ and $[-200, -64)$ corresponding to symbols $1, 2, 3$ and 4 respectively. The regions were determined by close inspection of the data and correspond to clusters of low and high positive/negative laser activity change.

We trained 5 RNNs with 2, 3, 4, 5 and 6 recurrent neurons having 3, 4, 5, 6 and 6 hidden non-recurrent neurons, respectively. All networks have 4 input and 4 output neurons corresponding to the 4-symbol alphabet. The 5 networks were chosen to represent a series of RNNs with increasing dynamical[12] and association[13] representational power. The training process consisted of 10 runs. Each run involved random initialization of RNN with small weights and 100 training passes through the training sequence $S = s_1 s_2 ... s_N$ (training epochs)[14]. After the first, second, fifth and every tenth epoch, we let the networks generate sequences $S(RNN)$ of length equal to the length of the training sequence S and computed the Lempel-Ziv entropy and cross-entropy estimates $h_{LZ}(S(RNN))$ (eq. (4)) and $h_{LZ}^{KL}(S|S(RNN))$ (eq.(6)), respectively.

Due to finite sequence length effects, the estimates of the cross-entropy $h_{LZ}^{KL}(S|G)$ between the training and model generated sequences S and G, respectively, may be negative. Since the only non-constant term in $h_{LZ}^{KL}(S|\cdot)$ is the number of cross-phrases in the training sequence S with respect to the model generated sequence, we use $c(S|\cdot)$ as the relevant performance measure. The higher is the number of cross-phrases, the bigger is the estimated statistical distance between the training and model generated sequences.

Figure 2 brings a summary of the average entropic measures (together with standard deviations) across 10 training runs. The entropy estimates $h_{LZ}(S(RNN)$ of the model generated sequences are shown with solid lines. The numbers of cross-phrases $c(S|S(RNN))$ (scaled down by a factor 10^{-3}) are shown with dashed lines. The horizontal dotted line in each graph corresponds to the training sequence entropy estimate $h_{LZ}(S)$. Simpler networks with 2 and 3 recurrent neurons tend to overestimate the training sequence entropy by developing relatively simple dynamical regimes (for each input symbol, attractive fixed points, or period-two orbits). This is also reflected by large sets of Lempel-Ziv cross-phrases (large $c(S|S(RNN))$). On the other hand, RNNs with 4, 5 and 6 recurrent neurons were able to develop in their state space increasingly sophisticated dynamical representations of the temporal structure in S. However, besides lower cross-entropy estimates $h_{LZ}^{KL}(S|S(RNN))$, a consequence of a more powerful potential for developing dynamical scenarios is a greater liability to over-complicated solutions that underestimate the training sequence entropy.

Next, for each number $i = 2, 3, .., 6$ of recurrent neurons we selected the best performing (in terms of cross-entropies $h_{LZ}^{KL}(S|S(RNN))$) network representative RNN_i. The representatives were then reformulated (via the extraction procedure described in the previous section) as finite state stochastic machines M_{RNN_i} and $M_{RNN_i(S)}$. The parameters T_1 (number of pre-test steps) and T_2 (number of

[11] `http://www.cs.colorado.edu/~andreas/Time-Series/SantaFe.html`

[12] input + current state $\rightarrow$ next state

[13] input + current state $\rightarrow$ output

[14] The weights were initiated in the range $[-1, 1]$, each coordinate of the initial state R^1 was chosen randomly from the interval $[-0.5, 0.5]$. We trained the networks through RTRL with learning and momentum rates set to 0.15 and 0.08, respectively.

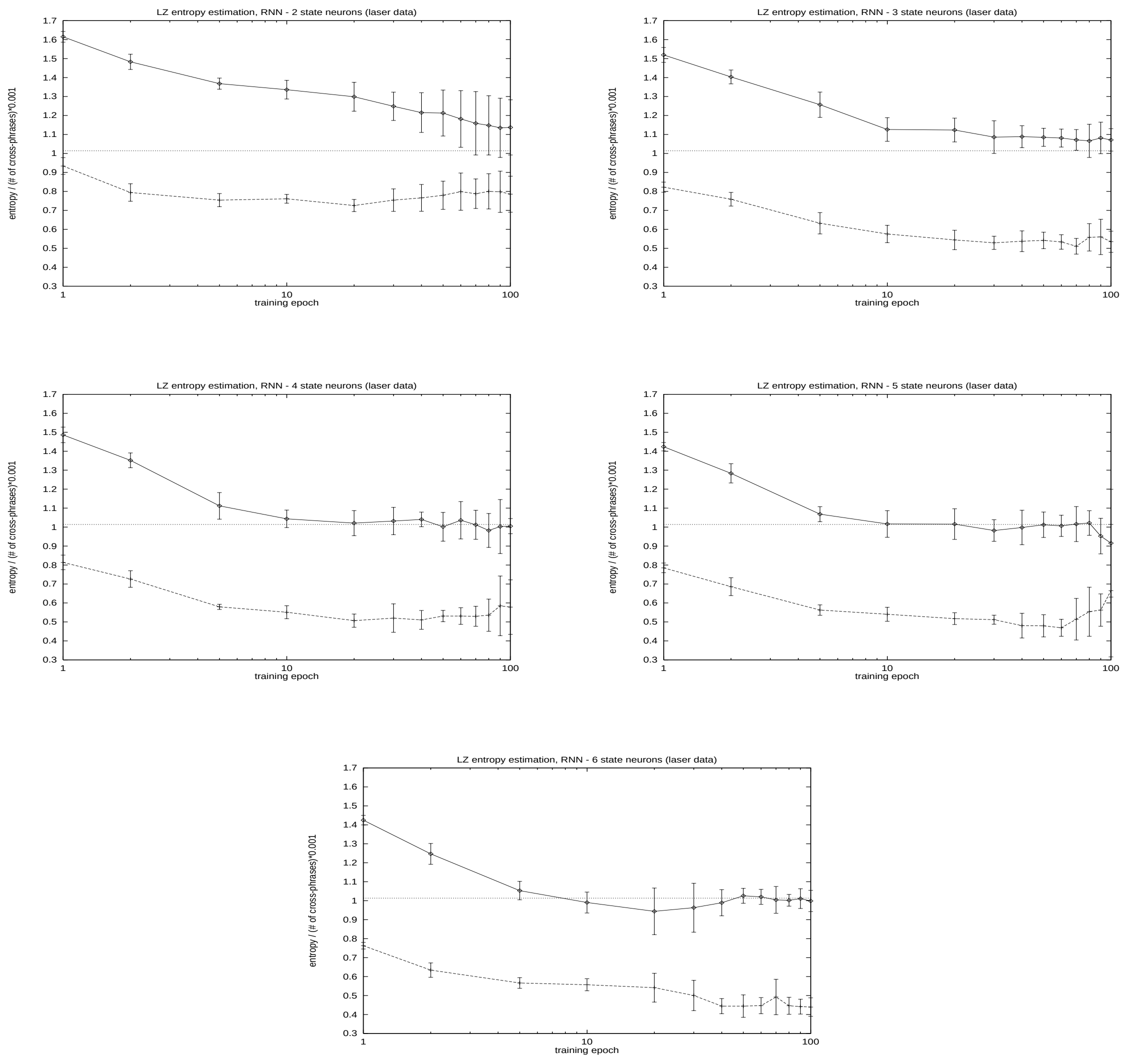

Figure 2: Training RNNs on the laser sequence. Shown are the average (together with standard deviations) Lempel-Ziv entropy estimates (solid lines) and numbers of cross-phrases (dashed lines, scaled by 10^{-3}) of the RNN generated sequences at different stages of the training process. Each graph summarizes results across 10 training runs.

test steps) of the extraction procedure were set to $T_1 = 10$, $T_2 = N$ and $T_1 = N-1$, $T_2 = 0$ for constructing the machines M_{RNN_i} and $M_{RNN_i(S)}$, respectively.

By running the DCS quantization algorithm we obtain for each network representative RNN_i, $i = 2, 3, ..., 6$, a series of extracted machines with increasing number of states. As in case of RNN training, we monitor the extraction process by letting the machines generate sequences G of length N (length of the training sequence S) and evaluating the entropy estimates $h_{LZ}(G)$ and $h_{LZ}^{KL}(S|G)$. In figure 3 we show the average (across 10 sequence generations) entropies (solid lines) and numbers of cross-phrases (dashed lines, scaled by a factor 10^{-3}) associated with the sequences generated by the extracted finite state machines M_{RNN} (squares) and $M_{RNN(S)}$ (stars). Standard deviations are less than 3% of the reported values. The upper and lower horizontal dotted lines correspond to the average entropy and scaled number of cross-phrases estimates $h_{LZ}(S(RNN)$ and $c(S|S(RNN)))$, respectively, computed on sequences generated by the RNN representatives. The short horizontal dotted line between the two dotted lines appearing on the right hand side of each graph signifies the training sequence entropy $h_{LZ}(S)$.

Four observations can be made.

1. For simpler networks (with 2 and 3 state neurons) that overestimate the training sequence stochasticity (entropy), the extracted machines $M_{RNN(S)}$ tend to overcome the overestimation phenomenon and produce sequences the entropies of which are close to the training sequence entropy.

2. The cross-entropies $h_{LZ}^{KL}(S|S(M_{RNN(S)}))$ corresponding to the sequences $S(M_{RNN(S)})$ generated by the machines $M_{RNN(S)}$ are systematically lower than the cross-entropies $h_{LZ}^{KL}(S|S(M_{RNN})$ associated with the machines M_{RNN}.

3. With respect to the Lempel-Ziv entropy measures h_{LZ}, d_{LZ}^{KL}, increasing the number of states in the machines M_{RNN} leads to formation of "faithful" finite state representations of the original RNNs. The sequences produced by both the mother recurrent network and the extracted machines yield almost the same entropy and cross-entropy estimates.

4. Finite state representations M_{RNN} of larger RNNs with higher representational capabilities need more states to achieve the performance of their network originals. The network RNN_2 with 2 recurrent neurons can be described with machines M_{RNN_2} having at least 30 states. To describe the representative network RNN_6 with 6 recurrent neurons one needs machines M_{RNN_6} with over 200 states.

For a deeper insight into the block distribution in sequences generated by the RNNs and their finite state representations we computed the entropy spectra of the training and model generated sequences. Each RNN representative RNN_i, $i = 2, 3, ..., 6$ and the corresponding machines M_{RNN_i}, $M_{RNN_i(S)}$ with 50, 100 and 200 states were used to generate 10 sequences of length equal to the length of the training sequence and the spectra (eq. (2)) were computed for block lengths 1,2,...,10 and inverse temperatures ranging from -60 to 120. As an example, we show in figure 4 the entropy spectra based on 6-block statistics of the training sequence and sequences generated by the machines M_{RNN_3}, $M_{RNN_3(S)}$ with 100 states extracted from the RNN representative with 3 recurrent neurons. Flat regions in the negative temperature part of the spectra appear because the sequences are too short to reveal any significant probabilistic structure in low probability 6-blocks. The machine $M_{RNN_3(S)}$ outperforms the machine M_{RNN_3} in mimicking the distribution of rare 6-blocks in the training sequence. Positive temperature part of the spectra show that both machines overestimate the probabilistic structure in the training sequence high probability 6-blocks. The set of highly prob-

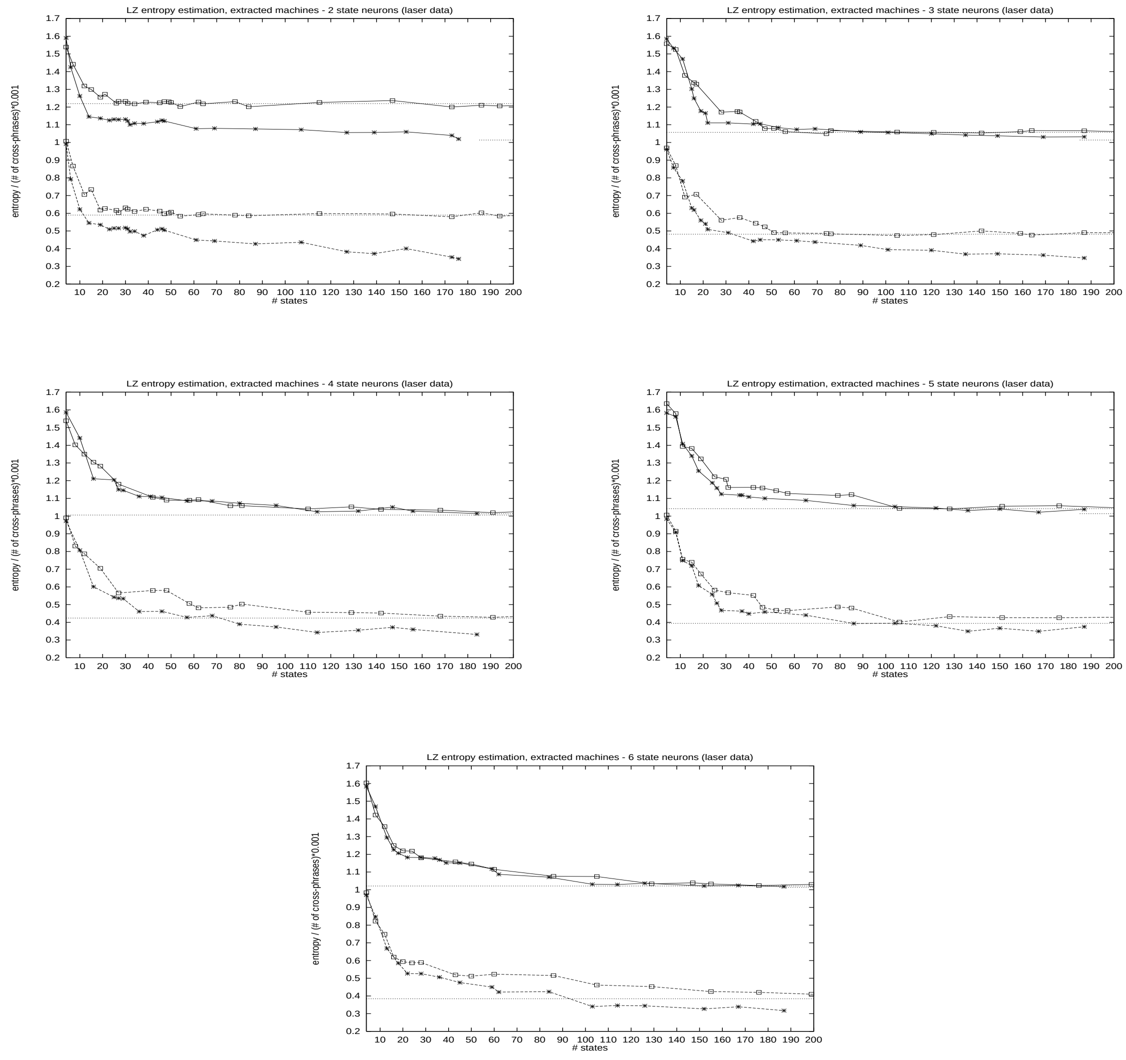

Figure 3: Lempel-Ziv entropy and cross-phrases performance of the machines extracted from the RNN representatives trained on the laser sequence S. Shown are the average (across 10 sequence generations) entropies (solid lines) and numbers of cross-phrases (dashed lines, scaled by a factor 10^{-3}) associated with the sequences generated by the machines M_{RNN} (squares) and $M_{RNN(S)}$ (stars). Standard deviations are less than 3% of the reported values. The upper and lower horizontal dotted lines correspond to the average entropies and scaled numbers of cross-phrases $h_{LZ}(S(RNN_i)$ and $c(S|S(RNN_i)))$, respectively, computed on sequences generated by the RNN representatives. The short horizontal dotted line appearing between the two dotted lines on the right hand side of each graph signifies the training sequence entropy $h_{LZ}(S)$.

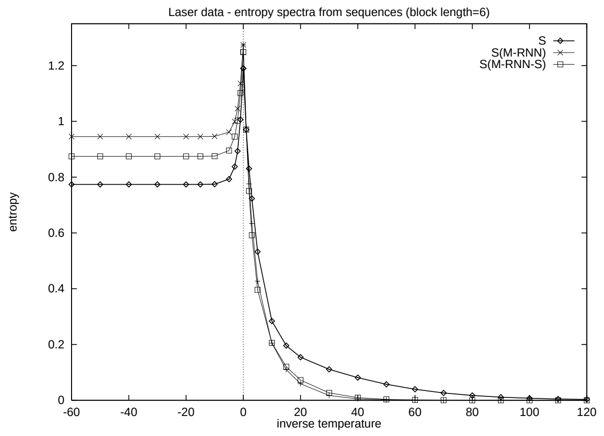

Figure 4: Estimation of entropy spectra (based on 6-block statistics) of sequences generated by the machines M_{RNN_3} (crosses), $M_{RNN_3(S)}$ (boxes) with 100 states extracted from the 3-recurrent-neuron representative RNN_3. The bold line with diamonds corresponds to spectrum of the training sequence.

able 6-blocks in sequences generated by the machine $M_{RNN_3(S)}$ is only slightly less constrained than that associated with the machine M_{RNN_3}.

Using the computed entropy spectra we can analyze the behavior of the studied models across various block lengths. For each model $\mathcal{M}$ and every block length $l = 1, 2, ..., L_{max} = 10$, we compare the entropy spectrum $h_{\beta,l}(S(\mathcal{M}))$, $\beta \in \{0, 1, 2, 3, 5, 10, 15, 20, 30, 60, 100\} = B^+$ and $\beta \in \{-60, -30, -20, -15, -10, -5, -3, -2, -1, 0\} = B^-$, of the model generated sequence $S(\mathcal{M})$ with that of the training sequence S through distances $\mathcal{D}_l^-(S, S(\mathcal{M}))$, $\mathcal{D}_l^+(S, S(\mathcal{M}))$, where for any two sequences S_1, S_2 over the same alphabet

$$\mathcal{D}_l^-(S_1, S_2) = \sum_{\beta \in B^-} |h_{\beta,l}(S_1) - h_{\beta,l}(S_2)| \tag{17}$$

and

$$\mathcal{D}_l^+(S_1, S_2) = \sum_{\beta \in B^+} |h_{\beta,l}(S_1) - h_{\beta,l}(S_2)|. \tag{18}$$

The distances $\mathcal{D}_l^-(S_1, S_2)$ and $\mathcal{D}^+(S_1, S_2)$ approximate the L_1 entropy spectra distances $\int_{-\infty}^0 |h_{\beta,l}(S_1) - h_{\beta,l}(S_2)| dq(\beta)$ and $\int_0^\infty |h_{\beta,l}(S_1) - h_{\beta,l}(S_2)| dq(\beta)$ corresponding to low and high probability l-blocks respectively. The measure q is concentrated on high absolute values of temperature. Because of finite sequence length, the statistics corresponding to such temperatures are better determined than the low temperature ones.

For each block length $l = 1, 2, ..., 10$, we studied whether the entropy spectra distances $\mathcal{D}_l^-$ and $\mathcal{D}_l^+$ associated with the network representatives RNN_i, $i = 1, 2, ..., 6$ differ significantly from those associated with the extracted machines M_{RNN_i} and $M_{RNN_i(S)}$. Significance of the differences is evaluated using the (non-parametric) Mann and Whitney U-test [22] at significance level 5%. We also test for significance in differences between the entropy spectra distances associated with the two kinds of RNN finite state representations M_{RNN} and $M_{RNN(S)}$. Results of the significance tests are summarized in table 1. For each considered model couple we report the number of block lengths on which one model significantly outperforms the other one. The results are reported separately for low and high probability blocks.

In view of such detailed subsequence distribution analysis performed via the entropy spectra, there is a strong evidence that

# states	model pair	# of state neurons				
		2	3	4	5	6
50	M, N	6(0),3(0)	0(3),3(3)	5(2),2(7)	1(4),0(3)	4(3),0(4)
	T, N	10(0),8(1)	10(0),7(1)	10(0),8(0)	10(0),7(0)	10(0),5(2)
	T, M	10(0),8(1)	9(0),8(2)	10(0),10(0)	10(0),8(0)	10(0),8(1)
100	M, N	8(0),1(0)	9(0),5(2)	6(0),1(0)	8(0),2(0)	5(0),0(1)
	T, N	10(0),9(1)	10(0),10(0)	10(0),9(0)	10(0),6(0)	10(0),9(0)
	T, M	10(0),9(1)	10(0),10(0)	10(0),10(0)	10(0),6(0)	10(0),8(0)
200	M, N	7(0),3(1)	7(0),1(3)	5(0),0(3)	3(1),2(0)	8(0),1(0)
	T, N	10(0),10(0)	9(0),10(0)	10(0),10(0)	10(0),7(0)	10(0),9(0)
	T, M	10(0),10(0)	9(0),10(0)	10(0),10(0)	10(0),9(0)	10(0),9(0)

Table 1: Model comparison in the laser data experiment with respect to the entropy spectra distances between the training and model generated sequences (each model generated 10 sequences). The entropy spectra were calculated for block lengths $1, ..., 10$. Significance of differences between the distances is evaluated using the Mann and Whitney U-test with significance level 5%. For each model pair $X, Y \in \{N, M, T\}$ we report 4 numbers in the following form: $n_X^-(n_Y^-), n_X^+(n_Y^+)$. Here, n_X^- and n_X^+ are the numbers of block lengths where the model X significantly outperforms the model Y (i.e. has significantly lower entropy spectra distances) in the negative and positive temperature regimes, respectively. The numbers n_Y^- and n_Y^+ in the parenthesis have analogical meaning, this time reporting the cases where the model Y is significantly better than the model X. The models RNN, M_{RNN} and $M_{RNN(S)}$ are denoted by N, M and T, respectively.

1. although in general the finite state reformulations M_{RNN} of RNNs do lead to improved modeling behavior, the difference between RNNs and the machines $M_{RNN(S)}$ is much more pronounced in favor of the machines $M_{RNN(S)}$

2. the machines $M_{RNN(S)}$ outperform the machines M_{RNN}, especially on low probability subsequences (negative temperature regimes).

One should be careful not to overestimate the significance results, though. For each model, we plot in figure 5 a histogram of 4 values related to the average (across 10 sequence generations) entropy spectra distances (AESD) between the training and model generated sequences in the negative ($-$) and positive ($+$) temperature regimes. The 4 values in the histogram correspond to (left to right) the maximal (across block lengths 1,2,...,10) AESD$-$, minimal AESD$-$, maximal AESD$+$ and minimal AESD$+$.

Figure 5 tells us that the significance results for the machines $M_{RNN(S)}$ do indeed unveil substantial modeling improvement over the corresponding machines M_{RNN} and mother RNNs.

5.2 Logistic map

In the second experiment, we generated the training data by running the logistic map $F(x) = rx(1-x)$ for 15,000 iterations from the initial condition $x_0 = 0.3$. The control parameter r was set to the Misiurewicz parameter value $r \approx 3.9277370012867...$ [43]. We converted the trajectory $x_1 x_2 x_3...$ into a symbolic sequence S over the alphabet $\mathcal{A} = \{1, 2\}$ via the generating partition $\mathcal{P} = \{[0, 1/2), [1/2, 1]\}$. Symbols 1 and 2 correspond to intervals $[0, 1/2)$ and $[1/2, 1]$ respectively.

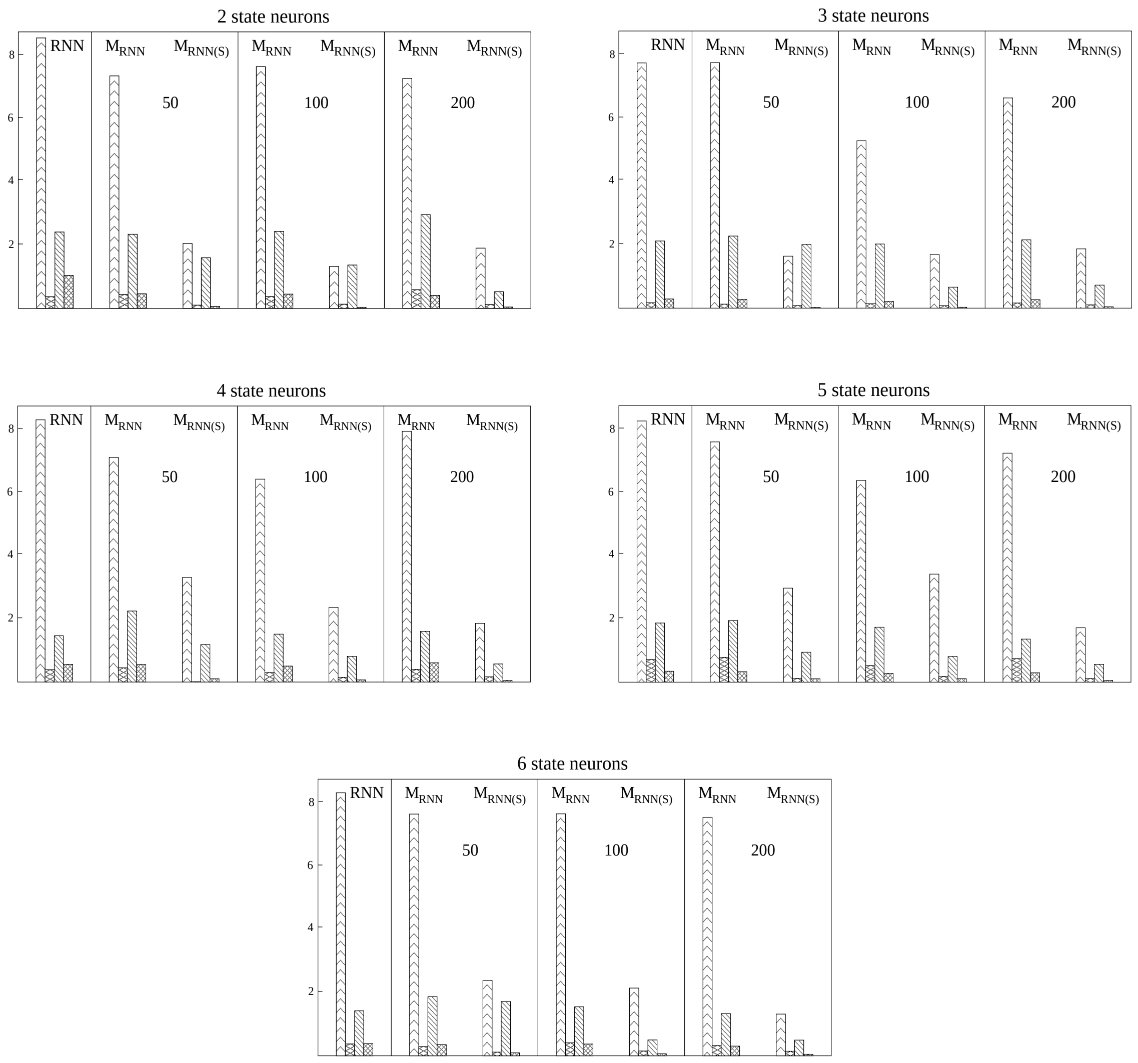

Figure 5: Average (over 10 sequence generations) entropy spectra distances (AESD) for RNNs and the corresponding extracted machines in the laser data experiment. The numbers inside the fields for couples of machines M_{RNN}, $M_{RNN(S)}$ signify the number of machine states. The four values shown for each model are (left to right) maximal and minimal (across block lengths 1,2,...,10) AESD in the negative temperature regime, maximal and minimal AESD in the positive temperature regime.

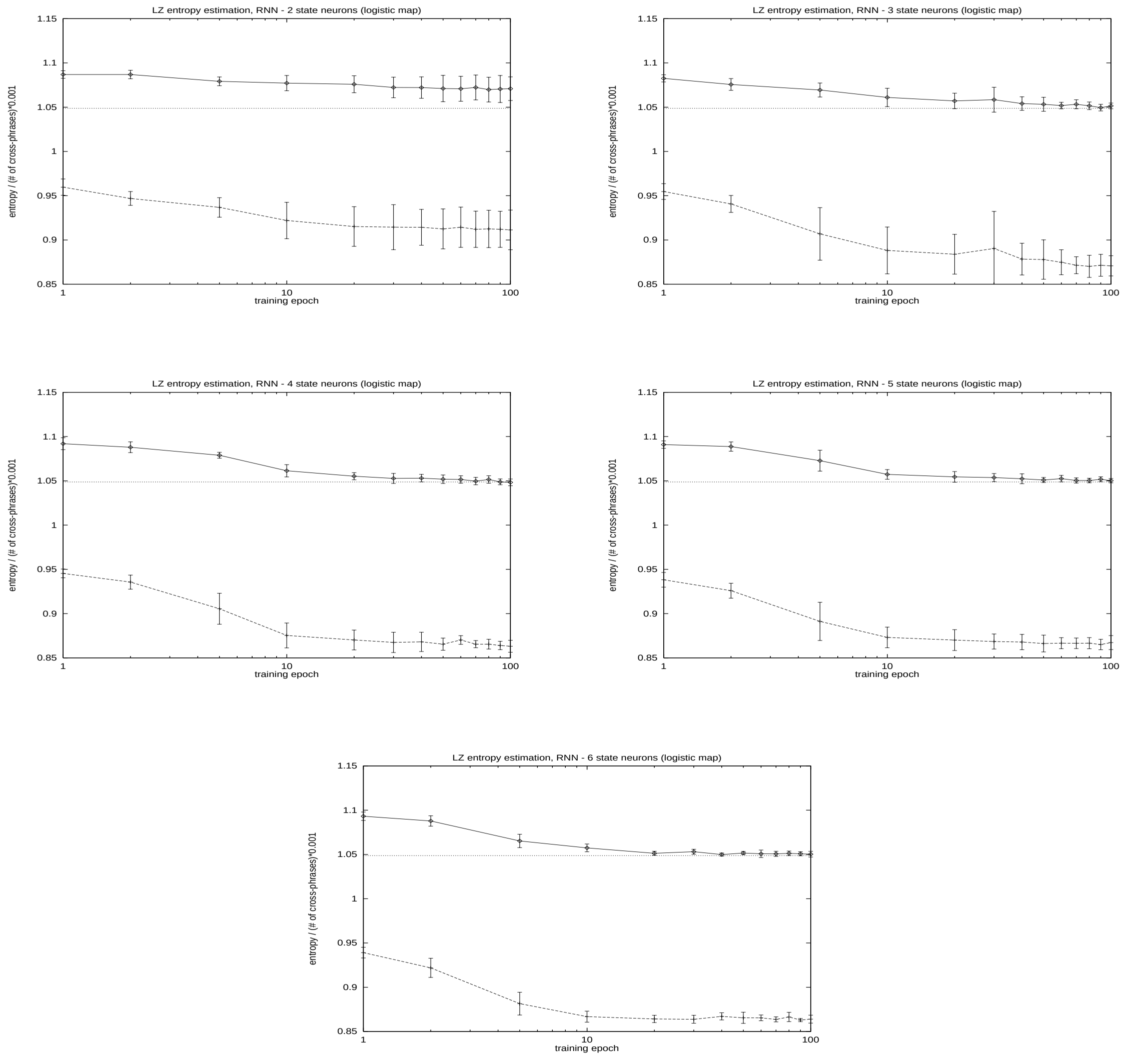

Figure 6: Training RNNs in the logistic map experiment. See caption to the figure 2.

As in the previous experiment, we trained 5 RNNs with 2, 3, 4, 5 and 6 recurrent neurons having 2, 3, 4, 5 and 5 hidden non-recurrent neurons, respectively. This time, every RNN had 2 input and 2 output neurons. All other training details are the same as in the laser data experiment.

Figures 6 and 7 are analogical to figures 2 and 3, respectively (the representative networks RNN_i, $i = 1, 2, ..., 6$ were selected as in the previous experiment).

Two things are worth noticing.

1. Only RNNs with 2 recurrent neurons overestimate the stochasticity of the training sequence by developing in their state space too simplistic dynamical regimes. Contrary to the previous experiment (see figure 2), more complex RNNs do not underestimate the training sequence stochasticity by developing too complex dynamical solutions. In fact, the networks with 4, 5 and 6 recurrent neurons exhibit almost the same Lempel-Ziv training scenarios.

2. The machines M_{RNN_i} extracted from *all* RNN representatives RNN_i, $i = 1, 2, ..., 6$ achieve the Lempel-Ziv entropy performance of their mother RNNs with less than 20 states. Moreover, compared with the laser data experiment, the difference in the the Lempel-Ziv performances between the machines M_{RNN_i} and $M_{RNN_{i(S)}}$, $i = 1, 2, ..., 6$ is less pronounced.

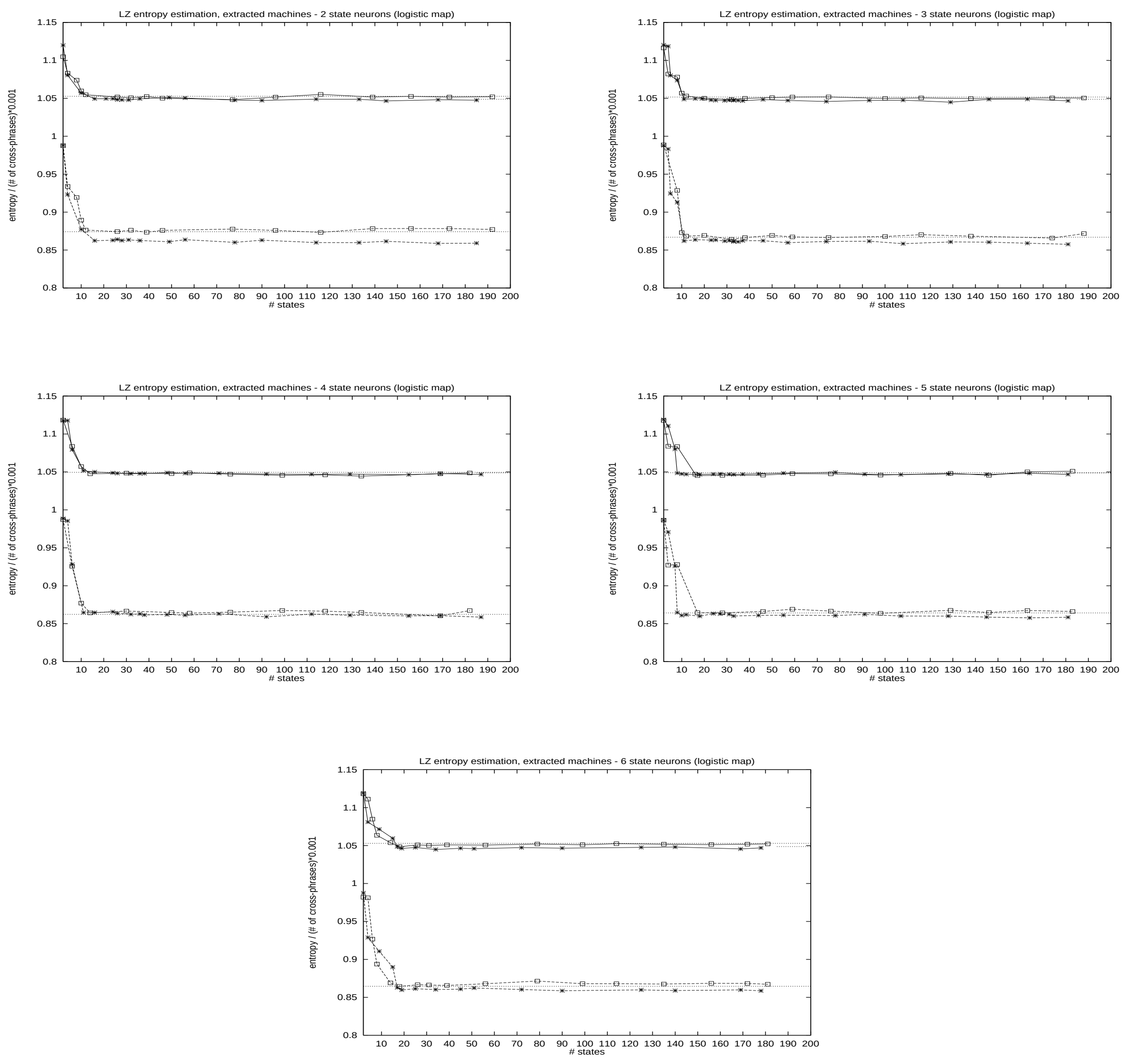

Figure 7: Logistic map experiment. Lempel-Ziv performance of the machines M_{RNN_i} and $M_{RNN_{i(S)}}$ extracted from the RNN representatives RNN_i, $i = 2, 3, ..., 6$. See caption to the figure 3.

These findings are in correspondence with the results obtained by running Crutchfield's ϵ-machine construction procedure [11, 12] on the laser and logistic map training sequences. The ϵ-machines M_ϵ were introduced as computational models of dynamical systems. They served as a means for determining the inherent computational structure of a given dynamical system [12, 9, 10], as well as its statistical momenta [11, 43]. One of the most notable results is the description of the formal relationship between entropy and complexity of a dynamical system on the edge of chaos [9], which has the insignia of a phase transition.

Briefly, the ϵ-machine construction on a data stream $S = s_1 s_2 ... s_N$ infers generalized states that give rise to the same set of possible future continuations. The states are identified using a sliding length-n window that advances through the sequence S one symbol at a time. A depth-n directed parse tree $\mathcal{P}$ is constructed by labeling links from the root to the leaves with individual symbols in n-blocks. This way, for each n-block in S, there is a path in $\mathcal{P}$ from the root to the corresponding leaf. The parse tree $\mathcal{P}$ is a compact representation of allowed n-blocks in S. For $l < n$, a depth-l subtree of $\mathcal{P}$, whose root is a node m is referred to as $\mathcal{P}^l_m$.

The stochastic machine $M_\epsilon = (Q, \mathcal{A}, \tau, \pi)$ is constructed as follows. Fix $l < n$. The set of states Q is the set of all distinct subtrees $\mathcal{P}^l_m$ in $\mathcal{P}$ [9]. By $[m]$ we denote the equivalence class of depth-l subtrees in $\mathcal{P}$, whose tree structure is identical to that of $\mathcal{P}^l_m$. The initial state is $[r]$, where r is the root of $\mathcal{P}$. There is an edge $[m_1] \rightarrow^s [m_2]$ in M_ϵ, if there is a link in $\mathcal{P}$ from p_1 to p_2, labeled by s, and $\mathcal{P}^l_{p_i} \in [m_i]$, $i = 1, 2$.

For each edge $[m_1] \rightarrow^s [m_2]$, the transition probability is computed as

$$\tau([m_1], [m_2], s) = \frac{N_{p_2}}{N_{p_1}}, \tag{19}$$

where N_k is the number of times the node k was passed through while parsing the sequence S and $\mathcal{P}^l_{p_i} \in [m_i]$, $i = 1, 2$, are such that the link from p_1 to p_2, labeled by s, is closest to the root r in $\mathcal{P}$. We refer the readers who wish to learn more about the ϵ-machines to [9].

There is a relatively simple computational structure in the Misiurewicz parameter logistic map data [43]. The ϵ-machine construction was not very sensitive to the choice of construction parameters – depth of the parse tree and depth of the subtrees functioning as state prototypes. Using depth-15 parse tree built on the logistic map training sequence and depth-3 state prototype subtrees we obtained a typical ϵ-machine with 4 states whose state-transition structure is identical to that of the ϵ-machine in [43] describing the computational properties of the logistic map at the Misiurewicz parameter value $r \approx 3.9277370012867....$

Laser data, on the other hand, show a rather complicated computational structure. The ϵ-machine construction turned out to be rather sensitive to the choice of construction parameters. Probably a much longer sequence would be needed to stabilize the construction procedure by identifying the relevant development modes in the sequence, or to resort to a higher level model in the hierarchy of computational models [10]. For example, using depth-3 subtrees as state prototypes, and parse trees of depth greater than 10, to obtain a generative computational model[15] of the laser sequence we needed to use the depth-27 parse tree leading to the ϵ-machine with 321 states.

In our experiments, the induced state transition structure in trained RNNs (and hence the state transition diagrams of the associated extracted machines) reflected the inherent computational structure of the training data by developing more computational states with more complicated state transition structure for the laser training data than for the logistic map data. In contrast with the laser data experiment, in the logistic map experiment the number of states needed to "faithfully"[16] refor-

[15] stochastic machine with at least one allowed state transition from every state

[16] with respect to the Lempel-Ziv entropy measures h_{LZ} and d^{KL}_{LZ}

# states	model pair	# of state neurons				
		2	3	4	5	6
16	M, N	1(8),4(3)	5(12),1(4)	2(7),1(0)	1(0),4(1)	2(0),0(0)
	T, N	17(2),12(0)	9(2),6(1)	8(7),1(1)	11(8),7(3)	13(6),6(6)
	T, M	16(3),16(0)	16(3),5(0)	11(2),1(0)	9(8),6(3)	14(6),6(9)
50	M, N	7(0),0(1)	0(1),1(1)	0(1),3(0)	1(1),4(0)	0(2),0(2)
	T, N	17(3),14(0)	19(0),5(5)	12(2),7(0)	13(2),13(0)	20(0),14(0)
	T, M	15(3),17(0)	19(0),2(3)	12(2),7(0)	13(1),10(0)	19(0),14(0)
150	M, N	11(0),0(4)	3(0),0(1)	2(1),1(13)	8(4),4(4)	6(1),0(0)
	T, N	16(2),11(0)	20(0),3(0)	12(2),5(0)	13(0),7(0)	20(0),17(0)
	T, M	16(2),18(0)	19(0),3(0)	12(2),16(0)	16(0),11(1)	18(0),15(0)

Table 2: Significance results corresponding to the Mann and Whitney U-test at 5% significance level for the entropy spectra distances on sequences generated by the induced models in the logistic map experiment. The entropy spectra distances were evaluated for block lengths $1, 2, ..., 20$.

mulate the mother RNN representatives as finite state stochastic machines M_{RNN} did not grow with increasing RNN representational power (compare figures 3 and 7).

As in the previous experiment, we used the idea of entropy spectra to perform a more detailed analysis of the block distribution in binary sequences generated by the induced models. The entropy spectra were computed for block lengths 1,2,...,20 and the entropy spectra distances were evaluated using the same sets of inverse temperatures B^- and B^+ (see eq. (17) and (18)) as in the laser data experiment.

We extracted the machines M_{RNN_i} and $M_{RNN_i(S)}$ with 16, 50 and 150 states from the representative networks RNN_i, $i = 1, 2, ..., 6$ and tested, as in the laser data experiment, for significance in differences between the entropy spectra distances associated with extracted machines M_{RNN}, $M_{RNN(S)}$ and mother RNNs. Table 2 is analogical to table 1 and reports, for the logistic map experiment, a summary of the significance tests performed with the Mann and Whitney U-test at 5% significance level.

Significance tests reported in table 2 confirm our previous findings that the finite state reformulations M_{RNN} of RNNs have in general better modeling performance than their network originals and that the machines $M_{RNN(S)}$ outperform both the mother RNNs and the corresponding machines M_{RNN}. By plotting histograms analogous to those in figure 5 we learned that, with respect to the detailed entropy spectra based subsequence distribution analysis, finite state representations $M_{RNN(S)}$ of RNNs lead to substantial modeling improvement over both the mother RNNs and the corresponding machines M_{RNN}.

6 Visualizing subsequence distribution

In [26] we formally studied[17] a geometric representation of subsequence structure, called the chaos game representation, originally introduced by Jeffrey [17] to study DNA sequences (see also [30, 35, 21]). The basis of the (generalized) chaos game representation of sequences over the alphabet $\mathcal{A} = \{1, 2, ..., A\}$ is an iterative function system (IFS) [3] consisting of A affine contractive maps[18]

[17] in a little bit more general context

[18] To keep the notation simple, we slightly abuse mathematical notation and, depending on the context, regard the symbols $1, 2, ..., A$, as integers, or as referring to maps on X.

$1, 2, ..., A$ with contraction coefficient $\frac{1}{2}$, acting on the D-dimensional unit hypercube[19] $X = [0, 1]^D$, $D = \lceil \log_2 A \rceil$:

$$i(x) = \frac{1}{2}x + \frac{1}{2}t_i, \quad t_i \in \{0, 1\}^D, \quad t_i \neq t_j \text{ for } i \neq j. \tag{20}$$

The chaos game representation $CGR(S)$ of a sequence $S = s_1 s_2 ...$ over $\mathcal{A}$ is obtained as follows [26]:

1. Start in the center $x_* = \{\frac{1}{2}\}^D$ of the hypercube X, $x_0 = x_*$.
2. Plot the point $x_n = j(x_{n-1})$, $n \geq 1$, provided the n-th symbol s_n is j.

If $A = 2^D$ for some integer D and S is a sequence generated by a Bernoulli source with equal symbol probabilities, the $CSR(S)$ "uniformly" samples X. If, on the other hand, some subsequences are not allowed in S, certain areas on X will stay untouched, suggesting a structure in S.

A useful variant of the chaos game representation, that we call the chaos n-block representation [26], codes allowed n-blocks as points in X. The chaos n-block representation $CBR_n(S)$ of a sequence S is constructed by plotting only the last points of the chaos game representations $CGR(w)$ of allowed n-blocks $w \in [S]_n$ in S.

Formally, let $u = u_1 u_2 ... u_n \in \mathcal{A}^n$ be a string over $\mathcal{A}$ and $x \in X$ a point in the hypercube X. The point

$$u(x) = u_n(u_{n-1}(...(u_2(u_1(x)))...)) = (u_n \circ u_{n-1} \circ ... \circ u_2 \circ u_1)(x) \tag{21}$$

is considered a geometrical representation of the string u under the IFS (20). For a set $Y \subseteq X$, $u(Y)$ is then $\{u(x) |\ x \in Y\}$.

Given a sequence $S = s_1 s_2 ...$ over $\mathcal{A}$, the chaos n-block representation of S is defined as a sequence of points

$$CBR_n(S) = \left\{S_i^{i+n-1}(x_*)\right\}_{i \geq 1}, \tag{22}$$

containing a point $w(x_*)$ for each n-block w in S. The map $w \to w(x_*)$ is one-to-one. Note that in this notation, the original chaos game representation of the sequence S can be written as

$$CGR(S) = \left\{S_1^{i+n-1}(x_*)\right\}_{i \geq 1}. \tag{23}$$

We proved in [26] several useful properties of the chaos n-block representations[20].

1. Let the distance between any two equally long sequences $\{x_i\}_{i \geq 1}$ and $\{y_i\}_{i \geq 1}$ of points in $\Re^D$ be defined as

$$d_s(\{x_i\}, \{y_i\}) = \sup_i d_E(x_i, y_i),$$

where d_E is the Euclidean distance. For large enough n, up to points associated with the initial n-block, the n-block representations $CBR_n(S)$ closely approximate the original chaos game representations $CGR(S)$. More precisely, denote by $CGR_n(S)$ the sequence $CGR(S)$ without the first $n-1$ points. Then,

$$d_S(CBR_n(S), CGR_n(S)) \leq \frac{1}{2^n}\sqrt{D}.$$

[19] for $x \in \Re$, $\lceil x \rceil$ is the smallest integer y, such that $y \geq x$

[20] One of the anonymous reviewers made an interesting comment that there are other choices for the maps of the underlying IFS mapping the n-blocks over the alphabet $\mathcal{A} = \{1, 2, ..., A\}$ to points in an Euclidean space so that the useful properties that we list here continue to hold with only slight modifications. For example we could use a one-dimensional IFS with the maps i defined as $i(x) = x/A + t_i/A$, $t_i = i - 1$. For a review of this kind of IFSs and an analysis of their computational power see [25].

Also, the chaos n-block representation $CBR_n(S)$ includes the "lower-order" representations $CBR_m(S)$, $m < n$, as its (more or less crude) approximations: Let $CBR_{m,n}(S)$ denote the sequence $CBR_m(S)$ without the first $n - m$ points. Then,

$$d_S(CBR_{m,n}(S), CBR_n(S)) \leq \frac{1}{2^m}\sqrt{D}.$$

2. The chaos n-block representation codes the suffix structure in allowed n-blocks in the following sense: if $v \in \mathcal{A}^+$ is a suffix of length $|v|$ of a string $u = rv$, $r, u \in \mathcal{A}^+$, then $u(X) \subset v(X)$, where $v(X)$ is a D-dimensional hypercube of side length $1/2^{|v|}$. Hence, the longer is the common suffix shared by two n-blocks, the closer the n-blocks are mapped in the chaos n-block representation $CBR_n(S)$.

3. The estimates of Rényi generalized dimension spectra [24] quantifying the multifractal scaling properties of $CBR_n(S)$, directly correspond to the estimates of the Rényi entropy rate spectra [34] measuring the statistical structure in the sequence S. In particular, for infinite sequences S, as the block length n grows, the box-counting fractal dimension [3] and the information dimension [4] estimates of the chaos n-block representations $CBR_n(S)$, tend exactly to the sequences' topological and metric entropies h_0 and h_1 (eq. (3)), respectively.

We use two-dimensional chaos 20-block representations of sequences generated in the laser data experiment to learn more about both the RNN training and the RNN finite state reformulation strategy. Translation parameters of the IFS (20) are set to $t_1 = (0,0)$, $t_2 = (1,0)$, $t_3 = (0,1)$ and $t_4 = (1,1)$.

Chaos 20-block representation of the training sequence S is shown in the upper left corner of figure 8. It is well-structured with dense clusters of points corresponding to allowed blocks in S. Empty areas around the corners $t_2 = (1,0)$ and $t_4 = (1,1)$ corresponding to symbols 2 and 4, respectively, reveal that blocks containing many consecutive 2s or 4s (monotonic series of large laser activity increases or decreases) are forbidden in S. Empty region around the line connecting the vertices t_2 and t_4 tells us that large blocks consisting only of symbols 2 and 4 cannot be found in S (laser does not produce large activity changes without interleaving periods of lower activity). On the other hand, there are large blocks in S consisting predominantly of symbols 1 and 3 as manifested by the concentration of points around the line connecting the vertices t_1 and t_3. Of course, the structure of allowed/not-allowed blocks in the laser sequence S is much more complicated and cannot be fully explained by simple rules such as those outlined above. One can nevertheless continue to explore the spatial structure of points in the chaos block representation of S to a much greater detail by concentrating on still smaller areas. The point we want to make here is that we propose a compact visual representation of allowed blocks in the studied sequences and this representation is not an ad-hoc heuristic, but

1. it is well-founded in the multifractal theory and statistical mechanics of symbolic sequences. The block representation reflects in its spectrum of generalized dimensions the statistical structure of allowed subsequences measured through the Rényi entropy spectra

2. the position of points is such that one can start with description of short allowed blocks by coarse analysis of the chaos block representation and then proceed to larger allowed blocks (with possibly more complicated structure) by analyzing the chaos block representation on finer scales.

In the upper right corner of figure 8 we show the chaos 20-block representation of the sequence $S(RNN)$ generated by a 3-state-neuron RNN (that became the representative network RNN_3 after 80 training epochs) initiated with small weights before training. RNNs with small recurrent weights have

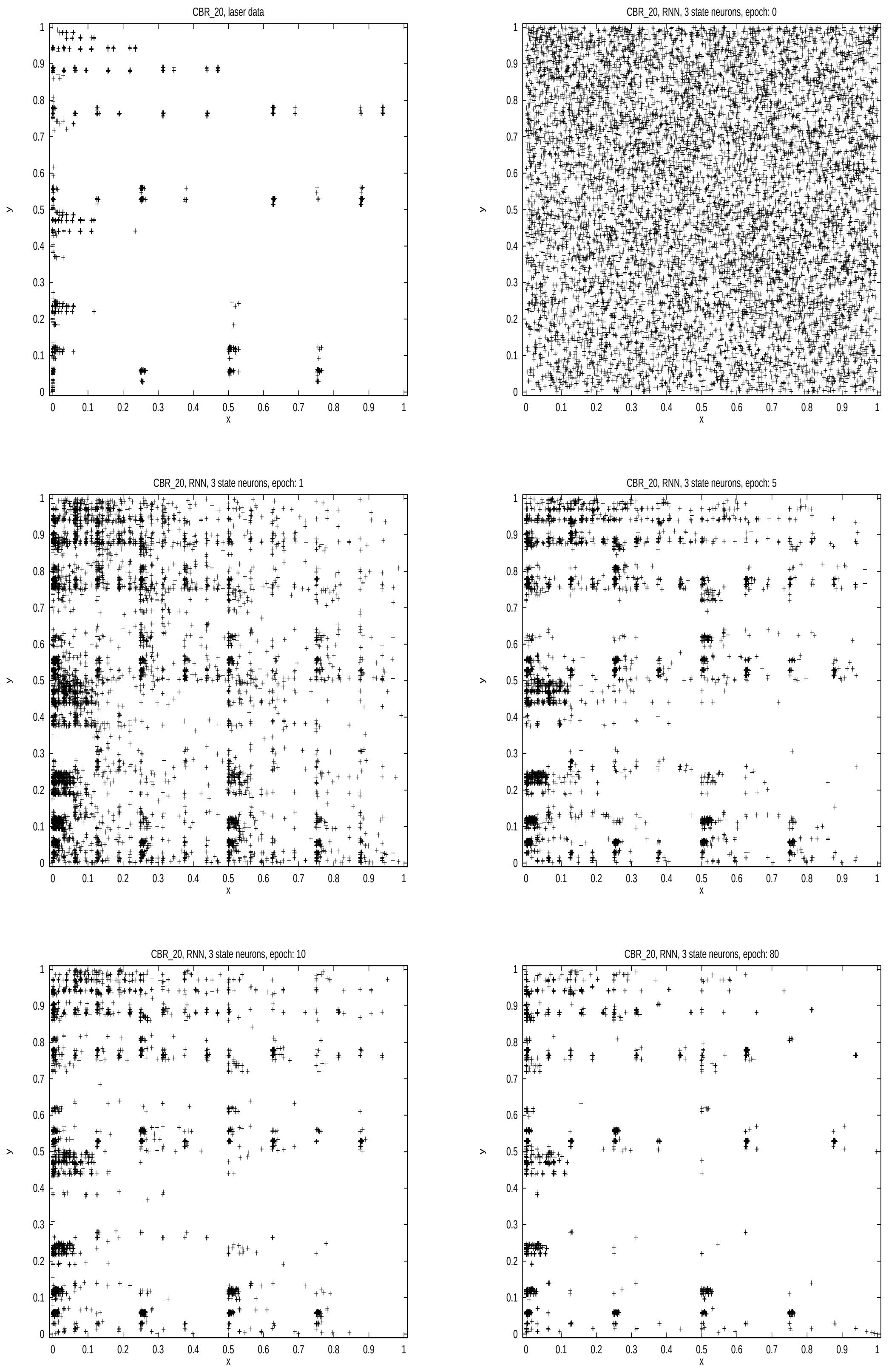

Figure 8: Chaos 20-block representations of the laser training sequence (upper left) and sequences generated by the 3-state-neuron-RNN before the training (upper right) and after the 1st, 5th, 10th and 80th epochs (central left to lower right).

a single attractive fixed point near the center of the recurrent neurons' activation space [28]. Small non-recurrent weights drive all the RNN outputs close to $1/2$ and so the distribution of symbols given by the RNN output is usually close to uniform. Hence, the initial RNNs can usually be well-described by a stochastic machine with equiprobable symbol loops (i.e. by a Bernoulli source with equal symbol probabilities). As expected, the chaos 20-block representation of a sequence $S(RNN)$ generated by such a network covers the whole square $X = [0,1]^2$ (all blocks of length 20 are allowed).

During the first training epoch, the RNN learns mainly the unconditional symbol probabilities without inducing in its state space any non-trivial dynamical regimes. At this stage, for each input symbol there is still one attractive fixed point in the RNN state space. As the training proceeds, bifurcations take place in the dynamic part of the RNN and the structure of allowed blocks in the RNN generated sequences $S(RNN)$ becomes more and more constrained (see figure 8). After 80 epochs the chaos block representation $CBR_{20}(S(RNN))$ clearly resembles that of that training sequence. However, one can still find in $CBR_{20}(S(RNN))$ isolated points corresponding to generated blocks that are not allowed in the training sequence. This is a typical scenario that we have found when interpreting the training process through the chaos block representations of RNN generated sequences.

Next, we use our tool for visual representation of subsequence structures to monitor the process of finite state stochastic machine extraction from the trained 3-state-neuron network RNN_3. In the upper right corner of figure 9 we see the chaos 20-block representation $CBR_{20}(S(M_{RNN_3(S)}))$ of a sequence generated by a 4-state-machine $M_{RNN_3(S)}$ extracted from RNN_3. The machine itself is shown[21] in the upper left corner of figure 9. Each state of the machine $M_{RNN_3(S)}$ contains a specific symbol loop. Higher probability symbol loops in states 1 and 3 accentuate the symbols 1,3 and are responsible for dense regions around the corners t_1 and t_3 in the chaos block representation $CBR_{20}(S(M_{RNN_3(S)}))$. Probability of the loop in state 4 is rather low and the region around the corner t_4 is relatively sparse (as it should be). On the other hand, there is no ground for the densely populated corner t_2. It will be cleared out as the number of states in $M_{RNN_3(S)}$ increases. The basis for triangle-like shapes in $CBR_{20}(S(M_{RNN_3(S)}))$ is a combination of loops in states 1, 2, 3 and period 2 cycles between states 1, 2 and 1, 3. Moreover, the loops in states 1,2 and 3 cause self-similar copies of the triangles to point towards the vertices t_1, t_2 and t_3. The transition from state 2 to state 4 on symbol 4 drives the self-similar triangle structure to the convex hull of vertices $(0,1/2)$, $(1,1/2)$ and $(1,1)$.

More machine states in $M_{RNN_3(S)}$ result in better modeling of allowed subsequences of the training sequence S. The machine $M_{RNN_3(S)}$ with 22 states already mimics the block distribution in S better than its mother network RNN_3 (see the lower right corner of the figure 8). The machine $M_{RNN_3(S)}$ with 45 states generates sequences with the set of allowed blocks very similar to that of the training sequence. We observed analogical scenarios when visualizing finite state machine $M_{RNN(S)}$ extraction from other mother RNNs via chaos block representation of machine generated sequences. In general, compared with machines $M_{RNN(S)}$, the chaos block representations of sequences generated by the machines M_{RNN} were more similar to the block representations of sequences generated by the mother RNNs. The machines $M_{RNN(S)}$ brought substantial improvement in approximating the set of allowed blocks in the laser training sequence.

The experiments with visual monitoring of allowed blocks in model generated sequences confirms, on a topological level[22], our findings with respect to the information theory and statistical mechanics based measures – Lempel-Ziv (cross)entropy and entropy spectra, respectively. Finite state reformulations of trained RNNs do not degradate the modeling performance, on the contrary, especially in the case of machines $M_{RNN(S)}$, they lead to improved modeling of the topological and metric structures

[21]each state transition in the machine is labeled by a symbol (that drives that transition) and the corresponding transition probability

[22]taking into account allowed/non-allowed blocks without any reference to their probability

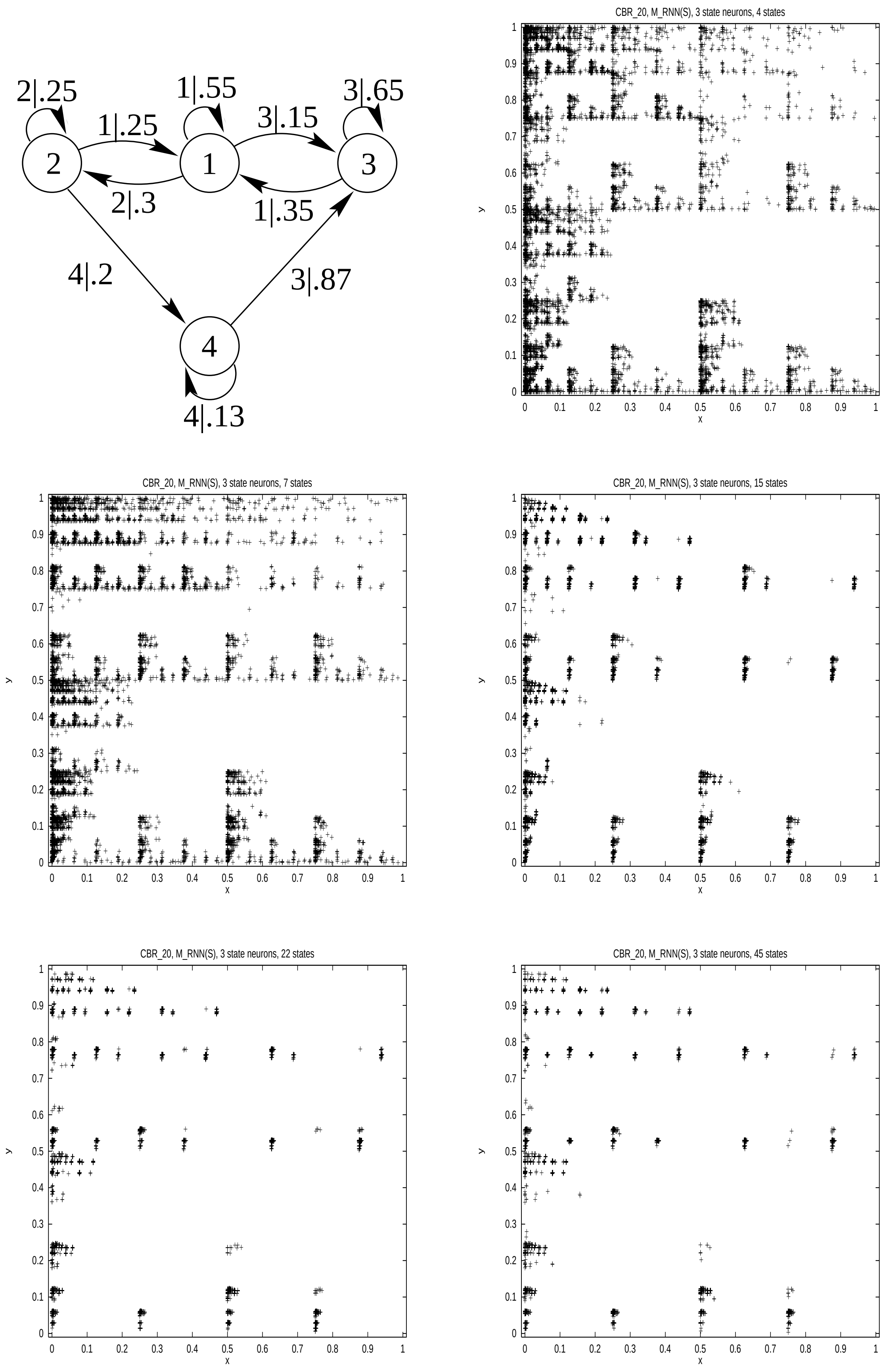

Figure 9: Chaos 20-block representations of sequences generated by the machines $M_{RNN_3(S)}$ extracted from the 3-state-neuron RNN representative RNN_3 trained on the laser sequence. Shown are the block representations corresponding to the machine with 4 states (upper right, the machine is shown in the upper left corner) and the machines with 7, 15, 22 and 45 states (central right to lower left).

in the training sequence.

In this paper we use the chaos block representation of a sequence S to represent the topological structure of allowed blocks in S. As mentioned earlier, in this case the fractal dimension of the block representation directly corresponds to the topological entropy h_0 (eq. (3)) of S. One could also code the block probabilities by a color of the corresponding points in the chaos block representation. Such a picture would visually represent in a compact way both the topological and metric structures of allowed subsequences in S. Moreover, analogically to the topological case, the information dimension of the block representation directly corresponds to the metric entropy h_1 of S.

7 Conclusion

The purpose of this paper was to investigate the knowledge induction process associated with training recurrent neural networks (RNNs) on single long chaotic symbolic sequences. When training the networks to predict the next symbol, the standard performance measures such as the mean square error on the network output are virtually useless. But it is our experience that RNNs still extract a lot of knowledge! This could be detected by considering the networks stochastic sources and letting them generate sequences which are then confronted with the training sequence via the information theoretic entropy and cross-entropy measures.

The second important issue that we addressed in this paper is a possibility of finite state compact reformulation of the knowledge extracted by RNNs during the training. We extracted finite state stochastic machines from RNNs and studied the sequences generated by such machines. We found that with sufficient number of states the machines do indeed replicate the entropy and cross-entropy performance of their mother RNNs. Moreover, machines constructed via the training sequence driven construction achieve even better modeling performance than their mother RNNs.

We performed the experiments on two sequences with different "complexities" measured by the size and state transition structure of the induced Crutchfield's ϵ-machines. The level of induced knowledge in a RNN can be assessed by the least number of states in a finite state network reformulation that achieves the RNN entropy and cross-entropy performance. We found that the level of induced knowledge in RNNs trained on the laser sequence with a lot of computational structure increased with increasing network representational power. On the other hand, all RNNs trained on a computationally simple logistic map (with the Misiurewicz parameter value) could be reformulated with relatively simple stochastic machines regardless of the RNN representational power. Hence, RNNs reflect the training sequence complexity in their dynamical state representations that can in turn be reformulated using finite state means.

Besides the standard information theoretic measures – entropy and cross-entropy – we use a statistical mechanical metaphor of entropy spectra to scan the subsequence distribution of the studied sequences across several block lengths and probability levels. Significance tests following a detailed model performance analysis through the entropy spectra of generated sequences revealed that the machines M_{RNN} extracted from RNNs in the autonomous test mode are only slightly better than the original RNNs. Furthermore, the significance tests revealed substantial improvement of the machines $M_{RNN(S)}$ over the mother RNNs and the corresponding machines M_{RNN}.

Visual representation of allowed block structure in the training and model generated sequences allowed for an illustrative insight, on the topological level, into both the RNN training and finite state stochastic machine extraction processes. It can be potentially used in a detailed analysis of the process of gradual confinement of the set of allowed subsequences generated by the extracted machines as the number of machine states increases. This way, one can identify state transition structures in the machines that are responsible for modeling the principal structure in allowed blocks of the training sequence, as well as the state transition structure responsible for "fine-tuning" the set of generated allowed blocks.

The visual representations of allowed blocks in symbolic sequences can be also used in building models of represented sequences. By quantizing the visual representations with a vector quantizer we identify as equivalent states blocks with long common suffices (because they lie in a close neighborhood) and subsequently construct the associated finite state stochastic machines [27]. Such machines can be shown to correspond to variable memory length Markov models [36].

Acknowledgments This work was supported by the VEGA grant 97/5195/605 of the Slovak Grant Agency and the Austrian Science Fund (FWF) within the research project 'Adaptive Information Systems and Modeling in Economics and Management Science" (SFB 010). The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Science and Transport.

References

[1] M.R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[2] F. Aurenhammer. Voronoi diagrams - survey of a fundamental geometric data structure. *ACM Computing Surveys*, (3):345–405, 1991.

[3] M.F. Barnsley. *Fractals everywhere.* Academic Press, New York, 1988.

[4] C. Beck and F. Schlogl. *Thermodynamics of chaotic systems.* Cambridge University Press, Cambridge, UK, 1995.

[5] G.E. Box and G.M. Jenkins. *Time Series Analysis, Forecasting and Control.* Holden Day, San Francisco, CA, 1970.

[6] J. Bruske and G. Sommer. Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, 7(4):845–865, 1995.

[7] M.P. Casey. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8(6):1135–1178, 1996.

[8] A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, 1(3):372–381, 1989.

[9] J.P. Crutchfield. Reconstructing language hierarchies. In H. Atmanspacher and H. Scheingraber, editors, *Information Dynamics*, pages 45–60. Plenum Press, New York, 1991.

[10] J.P. Crutchfield. The calculi of emergence: Computation, dynamics and induction. In *Physica D, special issue on the Proc. of the Oji Int. Seminar: Complex Systems, Namazu, Japan, 1993. (SFI tech. Rep. SFI 94-03-016)*, 1994.

[11] J.P. Crutchfield and K. Young. Inferring statistical complexity. *Physical Review Letters*, 63:105–108, July 1989.

[12] J.P. Crutchfield and K. Young. Computation at the onset of chaos. In W.H. Zurek, editor, *Complexity, Entropy, and the physics of Information, SFI Studies in the Sciences of Complexity, vol 8*, pages 223–269. Addison-Wesley, 1990.

[13] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[14] P. Frasconi, M. Gori, M. Maggini, and G. Soda. Insertion of finite state automata in recurrent radial basis function networks. *Machine Learning*, 23:5–32, 1996.

[15] C.L. Giles, C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun, and Y.C. Lee. Learning and extracting finite state automata with second–order recurrent neural networks. *Neural Computation*, 4(3):393–405, 1992.

[16] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison–Wesley, Redwood City, CA, 1991.

[17] J. Jeffrey. Chaos game representation of gene structure. *Nucleic Acids Research*, 18(8):2163–2170, 1990.

[18] A.I. Khinchin. *Mathematical Foundations of Information Theory.* Dover Publications, New York, 1957.

[19] T. Kohonen. The self–organizing map. *Proceedings of the IEEE*, 78(9):1464–1479, 1990.

[20] A. Lapedes, C. Barnes, C. Burks, R. Farber, and K. Sirotkin. Application of neural networks and other machine learning algorithms to dna sequence analysis. In G. Bell and T. Marr, editors, *Computers and DNA, SFI Studies in the Sciences of Complexity, volume VII*, pages 157–182. Addison-Wesley, 1989.

[21] W. Li. The study of correlation structures of dna sequences: a critical review. *Computer and Chemistry*, 21(4):257–272, 1997.

[22] H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Statist.*, 18, 1947.

[23] P. Manolios and R. Fanelli. First order recurrent neural networks and deterministic finite state automata. *Neural Computation*, 6(6):1155–1173, 1994.

[24] J.L. McCauley. *Chaos, Dynamics and Fractals: an algorithmic approach to deterministic chaos.* Cambridge University Press, 1994.

[25] C. Moore. Dynamical recognizers: real-time language recognition by analog computers. Technical Report Working Paper 96-05-023, to appear in Theoretical Computer Science, Santa Fe Institute, Santa Fe, NM, 1996.

[26] P. Tiňo. Spatial representation of symbolic sequences through iterative function system. *IEEE Systems, Man, and Cybernetics, part B: Cybernetics*, (in press), 1999.

[27] P. Tiňo and G. Dorffner. Constructing finite-context sources from fractal representations of symbolic sequences. Technical Report TR-98-18, Austrian Research Institute for Artificial Intelligence, Austria, 1998.

[28] P. Tiňo, B.G. Horne, C.L. Giles, and P.C. Collingwood. Finite state machines and recurrent neural networks – automata and dynamical systems approaches. In J.E. Dayhoff and O. Omidvar, editors, *Neural Networks and Pattern Recognition*, pages 171–220. Academic Press, 1998.

[29] P. Tiňo and J. Sajda. Learning and extracting initial mealy machines with a modular neural network model. *Neural Computation*, 7(4):822–844, 1995.

[30] J.L. Oliver, P. Bernaola-Galván, J. Guerrero-Garcia, and R. Román Roldan. Entropic profiles of dna sequences through chaos-game-derived images. *Journal of Theor. Biology*, (160):457–470, 1993.

[31] C.W. Omlin and C.L. Giles. Extraction of rules from discrete-time recurrent neural networks. *Neural Networks*, 9(1):41–51, 1996.

[32] J.C. Principe and J-M. Kuo. Dynamic modelling of chaotic time series with neural networks. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems, 7*, pages 311–318. MIT Press, 1995.

[33] J.C. Principe, A. Rathie, and J-M. Kuo. Prediction of chaotic time series with neural networks and the issue of dynamic modeling. *International Journal of Bifurcation and Chaos*, 2(4):989–996, 1992.

[34] A. Renyi. On the dimension and entropy of probability distributions. *Acta Math. Hung.*, (10):193, 1959.

[35] R. Roman-Roldan, P. Bernaola-Galvan, and J.L. Oliver. Entropic feature for sequence pattern through iteration function systems. *Pattern Recognition Letters*, 15:567–573, 1994.

[36] D. Ron, Y. Singer, and N. Tishby. The power of amnesia. *Machine Learning*, 25, 1996.

[37] J. Schmidhuber and S. Heil. Sequential neural text compression. *IEEE Transactions on Neural Networks*, 7(1):142–146, 1996.

[38] E.E. Snyder and G.D. Stormo. Identification of coding regions in genomic dna sequences: an application of dynamic programming and neural networks. *Nucleic Acids Research*, 21:607–613, 1993.

[39] J. Tani and N. Fukurama. Embedding a grammatical description in deterministic chaos: an experiment in recurrent neural learning. *Biological Cybernetics*, 72:365–370, 1995.

[40] E.A. Wan. Time series prediction by using a connectionist network with internal delay lines. In A.S. Weigend and N.A. Gershenfeld, editors, *Time Series Prediction*, pages 195–217. Addison–Wesley, 1994.

[41] R.L. Watrous and G.M. Kuhn. Induction of finite–state languages using second–order recurrent networks. *Neural Computation*, 4(3):406–414, 1992.

[42] A.S. Weigend, B.A. Huberman, and D.E. Rumelhart. Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1:193–209, 1990.

[43] K. Young and J.P. Crutchfield. Fluctuation spectroscopy. In W. Ebeling, editor, *Chaos, Solitons, and Fractals, special issue on Complexity*, 1993.

[44] J. Ziv and N. Merhav. A measure of relative entropy between individual sequences with application to universal classification. *IEEE Transactions on Information Theory*, 39(4):1270–1279, 1993.