Temporal Convolution Machines for Sequence Learning

Alan J. Lockett
Department of Computer Science
University of Texas at Austin
Austin, TX 78712
alockett@cs.utexas.edu

Risto Miikkulainen
Department of Computer Science
University of Texas at Austin
Austin, TX 78712
risto@cs.utexas.edu

Technical Report AI-09-04

Abstract

The Temporal Convolution Machine (TCM) is a neural architecture for learning temporal sequences that generalizes the Temporal Restricted Boltzmann Machine (TRBM). A convolution function is used to provide a trainable envelope of time sensitivity in the bias terms. Gaussian and multi-Gaussian envelopes with trainable means and variances are evaluated as particular instances of the TCM architecture. First, Gaussian and multi-Gaussian TCMs are shown to learn a class of multi-modal distributions over synthetic binary spatiotemporal data better than comparable TRBM models. Second, these networks are trained to recall digitized versions of baroque sonatas. In this task, a multi-Gaussian TCM performs effective sequence mapping when the input sequence is partially hidden. The TCM is therefore a promising approach to learning more complex temporal data than was previously possible.

1 Introduction

A major goal of artificial intelligence in general, and neural networks in particular, is to produce autonomous agents that can perceive and respond adaptively to real-world situations. While great progress has been made in training statistical models for analysis of static data in both generative and discriminative tasks, generation and recognition of time sequences has proven difficult. Success in this area is crucial to developing autonomous agents that operate in real-world, real-time tasks such as speech recognition, motion planning, and video and audio analysis. The most commonly used tool in sequence modeling remains the Hidden Markov Model (HMM) [1], despite the fact that HMMs require one unit per modeled state, making it prohibitively expensive to build large-scale temporal models using HMMs. The Temporal Restricted Boltzmann Machine (TRBM) can represent componential states and is thus exponentially less computationally intensive than an HMM with the same state space. However, while TRBM models are comparatively stronger than most of their competitors, further improvements to their accuracy would enable wider use in more real-world applications.

The difficulty with building statistical models of temporal data is in determining which data from the past is relevant. One common approach is to use a fixed time window in combination with a static learning model, converting temporal into spatial data. The window approach is insufficient because it cannot effectively account for the effect of information outside of the window. The other common approach is to develop a Markov model where the current time step of analysis depends exclusively on a fixed number of prior states. States that are outside the range of the Markov assumption can still be taken into account through a set of hidden variables that will in theory propagate relevant past information into future time steps for use when needed. TRBMs and Recurrent Neural Networks (RNNs) in general fall into this category. It has however proven difficult to develop learning algorithms that can make effective use of hidden variables to preserve state as necessary. The Temporal Convolution Machine (TCM) introduced in this paper focuses instead on learning to pick out correlations with past states directly using adaptively delayed links. This approach is shown in this paper to outperform TRBMs at the task of learning artificially generated spatiotemporal distributions, a task that is a relevant abstraction of more concrete real-world tasks such as visual or audio analysis. As a first example of such a task, TCMs are shown capable of reproducing a musical sequence, as well as mapping it

to harmony. The TCM is therefore a promising approach to learning more complex temporal data than was previously possible.

2 Foundations

This research is situated in the tradition of recurrent neural networks. Such networks with sufficient hidden elements can theoretically represent most temporal sequences of interest. A variety of algorithms exist for constructing and training recurrent neural networks, including Backpropagation Through Time [2], Long Short-Term Memory [3], and, more recently, Echo State Networks (ESNs) [4]. Generative models approach the sequence learning problem from an unsupervised rather than a supervised perspective (see e.g. [5],[6]). Among these, the TCM architecture proposed in this paper is a generalization of the Temporal Restricted Boltzmann Machine (TRBM) [7], itself an extension of the Restricted Boltzmann Machine (RBM) [8]. These models are based on undirected links, for which an efficient algorithm for training accurate models exists. Directed graphical models for sequence learning have also been explored, such as the Helmholtz Machine Through Time (HMTT) [9], but with less success. Models based on RBMs have the advantage that they can be stacked to form deep networks; for the RBM, these stacks have been named Deep Belief Nets (DBNs) [10]. The advantages of stacking are inherited by Temporal Convolution Machines, as will be discussed in section 3.3.

2.1 Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) [8] is an undirected graphical model with nodes divided into two sets, visible and hidden, subject to the restriction that nodes in the visible layer may only be connected to nodes in the hidden layer and vice versa. An RBM has Boltzmann-distributed joint probability

$$P(V,H) = \exp\left(V \cdot B^V + H \cdot B^H + H^T W V\right)/Z, \tag{1}$$

where V and H are (random) binary column vectors representing the state of the visible and hidden layers respectively, B^V and B^H are bias vectors for the visible and hidden layers, W is a matrix of weights connecting the visible to the hidden layer, and Z is a normalizing factor termed the partition function. The term inside the exponential is referred to as the energy function of the distribution. Specifying the energy function specifies the model completely. When the marginal probability of any individual node having state value 1 is computed, the result is a stochastic neuron with a logistic activation function. In order to obtain a sample from the model, Monte Carlo methods such as Gibbs sampling must typically be used.

RBMs are trained using gradient ascent on the maximum likelihood of the observed data. The gradient-based training rule for selecting W given a data source $\tilde{P}(V)$ and using $\langle\cdot\rangle$ to represent an ensemble average is

$$\Delta W \propto \left\langle HV^T \right\rangle_{P(H|V)\tilde{P}(V)} - \left\langle HV^T \right\rangle_{P(H,V)}. \tag{2}$$

The bias terms are trained similarly. A difficulty arises from the term on the right in that it cannot be computed directly but only estimated. Usually this estimation is accomplished by initializing V to the data generated by $\tilde{P}(V)$ and using the posterior distributions in a Markov chain to reestimate H and V successively until equilibrium is reached. Contrastive divergence (CD) can be used to reduce the number of sampling steps necessary [11]. CD estimates the expected value $\left\langle HV^T \right\rangle_{P(H,V)}$ by initializing the RBM with V drawn from $\tilde{P}$, sampling H from the posterior and then successively sampling V and H. In practice, it is more efficient to use the posterior probabilities in place of the sampled values for V and H during training, but the sampled values and not the probabilities must be used during each estimation step. CD is an approximate rule, but the performance improvement achieved by CD generally outweighs any error introduced by using this method.

2.2 Temporal Restricted Boltzmann Machine

The Temporal Restricted Boltzmann Machine (TRBM) [7] extends the RBM with time-varying biases so that B^H in the RBM model above is replaced with

$$B_t^H = L^H H_{t-1} + B^H, \tag{3}$$

where L^H is a set of lateral weights at the hidden layer. A specially trained initial bias B_0^H may also be used. The total probability of a temporal process $(V_t, H_t)_{t\in[0,\ldots,T]}$ is

$$P(H,V) = \prod_{t=1}^{T} P\left(H_t, V_t \mid H_0^{t-1}, V_0^{t-1}\right) \cdot P(H_0, V_0). \tag{4}$$

Essentially, then, a TRBM introduces a separate RBM for each time step, one representing each of the posterior distributions in the formula above. A problem arises when trying to sample from this model in that the product term

above includes all time steps up to T. While one would like to obtain a sample from the model by sampling successive posterior distributions, this procedure ignores the results of the future on the present via the distinct partition functions at each time step. However, this procedure can still be used as a to obtain an approximate sample of $P(V, H)$. This approach is used throughout the experiments in this paper.

The basic TRBM architecture can be extended in several ways [7], e.g. by incorporating multiple time delays, by adding time-varying biases on the visible nodes as well, and by encoding interactions with past visible states into b_t^H. Slight modifications can also be made to the TRBM model that allow for exact inference [12]. Real-valued deterministic neurons are substituted for the stochastic neurons in the hidden layer, with the caveat that standard Boltzmann learning must be replaced with a variant of the Backpropagation Through Time algorithm. However, even this solution can still require the network to learn a complex Markov sequence in the hidden layer.

3 Temporal Convolution Machines

The main problem with the TRBM is that hidden states must learn to encode intricate structures preserving relevant features of the past. The approach taken in the Temporal Convolution Machine (TCM) is instead to make better use of existing links by adding an adaptive sensitivity to prior states. Specifically, the TCM generalizes the TRBM by allowing the time-varying bias of the underlying RBM to be a convolution of prior states with any function. In this way, states in the TCM can depend directly upon arbitrarily distant past states. From a practical point of view, this effect can only be used to go one or two dozen states into the past, and hidden states are still needed to remember relevant past features. But the complexity that must be modeled in the hidden layer is nonetheless significantly reduced, resulting in a stronger models with comparable numbers of parameters.

3.1 The TCM Formalism

Formally, a TCM defines time-varying biases for an RBM by

$$\begin{aligned} B_t^H &= B^H + \sum_{\tau=0}^{t-1} F^H\left(t-\tau;\theta^H\right) V_\tau + \sum_{\tau=0}^{t-1} F^{LH}\left(t-\tau;\theta^{LH}\right) H_\tau, \text{and} \\ B_t^V &= B^V + \sum_{\tau=0}^{t-1} F^V\left(t-\tau;\theta^V\right) H_\tau + \sum_{\tau=0}^{t-1} F^{LV}\left(t-\tau;\theta^{LV}\right) V_\tau, \end{aligned} \tag{5}$$

where F^* are matrix-valued functions of time parameterized by θ^*. The notation F^* will be used to refer to any one of F^H, F^V, F^{LH}, or F^{LV}, and similarly for θ^*. The basic TRBM above would then be given by setting $F^{LH}(\tau) = \delta_1(\tau)L^H$ (where δ is the Kronecker delta) and setting the rest of the convolution functions to zero. The convolution functions provide a high level of control and adaptability to the network, since these functions can themselves be parameterized to obtain variable time sensitivities at individual nodes as needed.

The general purpose training rule for TCMs for a parameter θ^H of F^H is then

$$\Delta\theta_t^H \propto \sum_{\tau=0}^{t-1}\left[\left(H_t - \hat{H}_t\right) V_\tau^T\right] \odot \frac{\partial}{\partial\theta} F^H(t-\tau), \tag{6}$$

where $\odot$ represents componentwise multiplication of vectors or matrices, $\hat{H}_t$ is sampled from $P\left(H_t, V_t \mid V_0^{t-1}, H_0^{t-1}\right)$, H_t is sampled from $P\left(H_t \mid V_0^t, H_0^{t-1}\right)\tilde{P}\left(V_t \mid V_0^{t-1}, H_0^{t-1}\right)$, and similarly for V_t and $\hat{V}_t$. The learning rule for a general parameter θ^* of F^* is readily obtained by analogy from (6).

The class of TCMs is quite a large one. The convolution functions F^* can take whatever forms are found to be useful or efficient. including those dependent on the prior states of the network. The approach taken here chooses convolution functions with trainable but fixed forms, since these functions work reasonably well and utilize manageable numbers of parameters.

3.2 A Gaussian TCM

A Gaussian TCM (G-TCM) can be formulated by choosing a convolution function with a kernel of Gaussian form,

$$F^*(\tau) = W^* \frac{1}{\Sigma^*\sqrt{2\pi}} \exp\left(-\frac{1}{2\Sigma^{*2}}\left(\tau - M^*\right)^2\right), \tag{7}$$

where M^* and Σ^* are matrices containing the mean and variance respectively for each neural link, and all matrix operations are assumed to be componentwise with τ expanded as necessary to a matrix of required dimension with all entries equal to the scalar τ. The form of the convolution function is designed so that each link can detect correlations

at arbitrary time distance and can adjust the location and scale parameters to respond to temporal correlations. This approach works because the form of F^* is smooth, unimodal, and nonzero wherever W^* is nonzero. This form is intended for use when each link needs information from a particular time step or cluster of time steps at some fixed time in the past. Ideally, for appropriate problems, the scale parameters Σ^* will start out large and then gradually become small as the location parameter M^* becomes more accurate. Thus the G-TCM should be able to represent certain problem classes better than a TRBM with sensitivity to a fixed number of prior time steps n, because the links can look arbitrarily far into the past.

The drawback is that the G-TCM must look at all prior timesteps during training and during inference. While this is a significant drawback for computer simulation, it is not necessarily the case that a physical implementation would be overly slow or complex. Also, the sequence of prior states can be clipped as needed based on the particular parameters of the G-TCM. A more serious problem is that the G-TCM can represent can only represent strictly positive or strictly negative temporal correlations between each pair of nodes. Multiple Gaussian links can be used to fix this problem.

3.3 A Multi-Gaussian TCM

Rather than just having one time-delayed link, a Multi-Gaussian TCM (MG-TCM) allows a fixed number of Gaussian-delayed links to each node. Thus both positive and negative correlations between two nodes are possible, depending on the time variable. With K links, the convolution function for an MG-TCM is given by

$$F^*(\tau) = \sum_{k=0}^{K} W_k^* \frac{1}{\Sigma_k^* \sqrt{2\pi}} \exp\left(-\frac{1}{2{\Sigma_k^*}^2}\left(\tau - M_k^*\right)^2\right). \tag{8}$$

An MG-TCM can represent much more complex relationships than a G-TCM, but it retains the same benefits as a G-TCM such as smoothness and unimodality with respect to each parameter (though it is multimodal overall), with the downside that the extra parameters impose an additional computational burden. As the experiments will show, however, a MG-TCM can build a much more accurate probabilistic model than a G-TCM or a TRBM in certain cases.

3.4 Deep Stacked TCMs

As with Deep Belief Nets (DBNs) [10] and TRBMs [7], individual TCMs can be stacked to form deep networks. A single TCM is trained first, and then the parameters of that TCM are frozen. A second TCM is trained using the output of the first TCM as input. To simplify training, For the training in this paper, the neuron activation function on the first TCM can be converted from stochastic neurons to threshold neurons based on the computed probability at each node; this step stabilizes the sequences being presented to the next higher layer and was found experimentally to improve the overall outcome. The stacking process can be continued as desired to obtain a deep stack of the desired depth. To generate from a stacked model, one simply generates from the top TCM in the stack, and then propagates the visible layer downward through the lower stacks (again using thresholded neurons in this research).

All the same benefits that stacking provides to DBNs and TRBMs accrue to TCMs as well, since a TCM is a generalization of these models. For instance, it can be shown that under certain conditions, the probabilistic model encoded by the stack will more accurately model the training distribution. To achieve such accuracy, the initial distribution encoded by each subsequent TCM needs to be the same as that of the stack previously trained, but in practice good refinements can be trained without adhering strictly to this condition. The relevant theorems for RBMs and TRBMs are discussed in detail in [7] and [10], and the results apply to TCMs as well.

4 Experiments

Two experiments were run in order to demonstrate the validity of the TCM model. The first experiment examines the ability of the G-TCM and MG-TCM to learn randomized spatiotemporal distributions consisting of overlaid Gaussian forms; the performance of these models is compared to TRBMs with one and three delay taps. The second experiment examines the ability of an MG-TCN to learn real-world sequence mappings using baroque sonatas.

In training these networks, a momentum factor of 0.5 was used to average over the stochastic gradient to provide faster and smoother convergence of the network parameters. As with TRBMs [7], the learning rate was increased as the network performance improved. The need to increase the learning rate arises because the learning rules are based on prediction errors, and as these errors are reduced by training, it becomes useful to amplify the error signal so that the network moves more quickly towards a solution. Rather than doubling the learning rate at fixed intervals as in [6], a variable learning rate was adopted dependent on the prediction error at the visible layer, given by $\eta_t = -\left(\frac{1}{50}\right)\log\left(E_{\text{avg}}(t)\right)$, where $E_{\text{avg}}(t)$ is a moving average computed from the bit error percentage for reconstruction of the visible layer during the contrastive divergence step, initialized to 0.40 at the beginning of training.

The parameters for the TCMs were initialized as follows. All weights and biases were sampled independently from a standard normal distribution. The location parameters M^* for the Gaussians were sampled from a beta distribution

Table 1: Mean Squared Error Rates for Various Models over Ten Trials

Trial	1	2	3	4	5	6	7	8	9	10	Avg
G-TCM	.0540	.0265	.0964	.0248	.0231	.0356	.0462	.0291	.0450	.0284	.0414
MG-TCM	.0070	.0047	.0106	.0074	.0135	.0157	.0078	.0080	.0090	.0099	.0094
1-TRBM	.0555	.1588	.0697	.0743	.0517	.0964	.0298	.1033	.0582	.0519	.0750
3-TRBM	.0358	.0357	.0252	.0491	.0159	.0361	.0630	.0660	.0314	.0888	.0447

with $\alpha = 2$ and $\beta = 5$, multiplied by 10 to give location parameters in the range $[0, 10]$ with greater emphasis on location parameters closer to zero. The scale parameters Σ^* were initialized to a fixed value of 3. These parameters were optimized experimentally; the parameters of TRBMs used for comparison were optimized similarly. The TRBMs were given delayed links laterally in the visible and hidden layers and between the visible and hidden layers. Finally, the TCMs and TRBMs in these experiments did not have specially trained initial biases, since the networks performed better at these tasks without a separate initial bias.

4.1 Learning Spatiotemporal Distributions

The first experiment demonstrates that G-TCMs and MG-TCMs can learn a model of spatiotemporal probability distributions over binary-vector-valued processes. A 25×25 grid of probability values was generated to provide a target distribution for learning. Each element in the grid represents the probability of observing the "on" state, 1, at a particular neuron at a particular point in time independent of all other neurons and time steps.

The grids used in these experiments were generated by taking a composite of three two-dimensional unnormalized pseudo-Gaussians over space and time according to the formula

$$p(t,s) = \max_{k=0,1,2} \exp\left(-\frac{1}{2}\left((t,s)-\mu_k\right)\Sigma_k^{-1}\left((t,s)-\mu_k\right)^T\right). \tag{9}$$

The means μ_k were chosen to lie between 5 and 20 in each dimension, and the entries for the covariance matrices Σ_k were constrained to be less than 5. In addition, the average probability mass over the grid points was constrained to be between .06 and .20 to ensure a good balance of low and high probability points in the grid. Pseudo-Gaussians that did not fit this constraint were discarded during the generation phase. In many cases, Σ_k was not positive semidefinite, resulting in shapes with a hyperbolic rather than an elliptical, Gaussian form. These hyperbolic structures were used to increase the variety of shapes to be modeled. The probability grids generated using this process have strong local structure, but also can transition quickly from low to high probability areas (Figure 1, left side). These properties are realistic for the types of problems G-TCMs should address (such as perception of motion or audio signals), and this fact makes learning these distributions a good first test for the capabilities of G-TCMs.

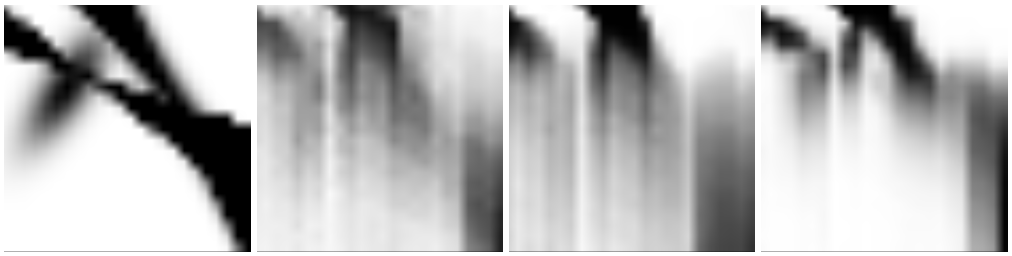
Training data for the G-TCM was sampled from the probability grid independently at each learning step. The resulting 25×25 binary grid was presented to a G-TCM with 25 visible and 25 hidden nodes as 25 successive 25-bit vectors. The G-TCM was trained with $5,000$ successive samples from the probability grid. The networks were trained online using contrastive divergence, with elements from the sample being presented one time step at a time. After training, 50 samples were drawn from the G-TCM using Gibbs sampling with 100 iterations. Rather than taking the binary values of the sample, the raw probabilities at the last step were used instead so that fewer samples were required; the binary values were used at all steps during sampling, however. These sample probabilities were averaged to produce an estimate of the probability distribution learned by the model, which can then be compared to the probability grid used for training to obtain a visual and numeric representation of the error.

The first G-TCM was then frozen, and a stack of three G-TCMs was developed by successively training two more G-TCMs using the same procedure. The results are shown in four heat map panels in Figure 1(a). The first panel is the probability grid on which the network was trained. Lighter values are closer to zero; darker values are closer to one. Time is laid out along the horizontal axis, while the vertical axis represents space. The second panel is the average of 50 samples drawn from the first layer of the G-TCM. The third and fourth panels respectively show the averaged samples drawn from the second and third layers of the stacked G-TCM. The one-layer G-TCM generally captures the shape of the distribution, but not accurately. By the second layer, however, the stack of G-TCMs takes on a reasonable representation of the distribution, which is then refined further by the third layer.

Models were subsequently trained for an MG-TCM with $K = 3$, a TRBM with a single delay tap, and a TRBM with three delay taps (3-TRBM) in order to compare the learning performance of all four models on this problem. The mean squared errors over ten separate trials on each are given in Table 1. The visual representations are perhaps most revealing; an example of all four methods learning the same distribution is shown in Figure 1; the other nine trials

(a) A Gaussian Temporal Convolution Network (G-TCM) learning a 25×25 spatiotemporal distribution

(b) A Multi-Gaussian Temporal Convolution Network (MG-TCM) with three gaussian delay taps



(c) A Temporal Restricted Boltzmann Machine (TRBM) with a single delay tap

(d) A Temporal Restricted Boltzmann Machine (3-TRBM) with three delay taps

Figure 1: Testing the ability of a various temporal probabilistic models to learn a spatiotemporal distribution. These panels represent a 25×25 binary distribution over space and time in which each cell is independent of the others. The vertical axis is space, the horizontal axis is time, and the shading of each cell represents the probability of that cell being in the "on" state, 1. The leftmost panel in each row contains the distribution being learned, generated as a composition of three pseudo-Gaussians. The second through fourth panels are the distributions learned by each model type. The second panel is generated from a single model, and the third and fourth panels add a second and third layer of the same model type respectively to form a stacked probabilistic model. The MG-TCM model builds a substantially closer model of the target distribution than the other approaches, the G-TCM is slightly better than the 3-TRBM, and the TRBM is substantially weaker. The exact performance measures are given in Table 1.

are generally comparable to those shown. The temporal striations visible in all the models seem to be a feature of the Boltzmann machine at the base of all these architectures. Overall, each model except for the TRBM with only one delay learned a reasonable model of the target distribution. The mean squared error rates show that the MG-TCM performs substantially better than the others, with .0094 average error. The G-TCM comes in second at .0414, and the TRBM with three delays achieves a close third at .0447. The TRBM with one delay came in last at .0750. A t-test shows that all of these comparisons are statistically significant with $p < .001$.

4.2 Learning to Play Baroque

Music is a good test domain for structured temporal sequences. In this experiment, a stack of three MG-TCMs with $K = 5$ was trained on digitized versions of five sonatas by baroque composer Domenico Scarlotti, namely, Sonatas No. 1, 20, 27, 159, and 531. These pieces were chosen because they exhibit strong counterpoint, nearly all notes have equal duration and there are few complex rests or rhythms. These features make a good target for initial experimentation.

A binary sequence representing the sonatas was built based on a midi representation. The sonatas were first normalized to the key of C major / A minor to simplify the task. Each time step was taken to represent a sixteenth note. A 128-bit vector was used for each time step, one bit for each of 128 notes available in the midi format. The "note on" events of the midi file were snapped to the closest time step, and the note contained in the event was marked to the "on" state, 1, in the bit vector for that time step. The first four measures (64 time steps) were taken from each of the sonatas. Out of the 128 possible notes, only 51 actually occurred in these pieces, so the bit sequence was truncated to 51 x 64. The first MG-TCM in the stack was trained with 500 repetitions of each sonata. The second and third MG-TCMs were trained with 250 repetitions. After training, the network was made deterministic by converting the stochastic neurons to threshold units without changing any parameters. Despite this change, the network still has to be sampled to equilibrium because of its undirected connections, but the outcome is deterministic and repeatable.

The resulting MG-TCM stack was tested by initializing the network with the first 16 time steps of each sonata, then generating the last 48 time steps. Over the five sonatas, these steps were generated with an average bit error rate of 5.96%. The MG-TCM was evaluated further by clamping the correct values for the right-hand melody to the network as input and reading off the left-hand harmony generated by the network with the upper half of the range clamped. In this case, the network is performing a sequence mapping task, where given an input process x_t (the right-hand melody) the goal is to produce a corresponding output process $y_t = f(x_0, \ldots, x_t)$ (the left-hand harmony). When run in this fashion, the first layer of the MG-TCM gives an average bit error rate of 1.4% over the five sonatas. The MG-TCM networks therefore proved to be highly robust in learning and mapping real-world sequential data.

5 Discussion and Future Work

The results show that TCMs improve the sequence-learning abilities of TRBMs significantly. Not only can several sequences be reproduced probabilistically, but they can also be mapped to other sequences. The results of the last experiment are particularly interesting because signal completion is an important issue in real-world problems. For example, in real-world speech recognition, a pure speech signal is almost never encountered; the source signal almost always needs to be separated from the background first, and sometimes background noise masks whole frequency bands. Source separation and denoising are close relatives of the sequence mapping problem described above, which makes the MG-TCM a promising starting point for developing complex applications.

The method can also be developed further. For instance, it would be worthwhile to search for convolution functions with even more desirable characteristics than those presented in this paper. For example, convolution functions that can be expressed using recurrent definitions could maintain a running sum rather than having to compute the convolution separately at each time step. If recurrent functions with robust time-sensitivity envelopes could be found, a TCM based on such functions would be much faster for training and use, though it is not immediately clear that such functions exist.

Although the convolution functions explored in this research do not depend on past states, such an architecture (mathematically similar to TCMs) was proposed in [13] in the context of hierarchical Echo State Networks. The drawback of such networks is that the number of connections is typically cubic in the size of the network, causing severe memory issues even for moderately sized networks. However, it may be possible to limit or constrain the form of such a convolution so that it becomes computationally feasible, in which case TCMs with variable time envelopes during inference could prove very powerful indeed.

6 Conclusion

Statistical modelling of temporal sequences is necessary in several important problems, including phoneme recognition, real-time visual processing, and signals processing. The Temporal Convolution Machine provides a flexible framework for developing gradient-based training algorithms for temporal generative models, based on recent ad-

vances in Deep Belief Nets and Temporal Restricted Boltzmann Machines trained with contrastive divergence. The Gaussian and Multi-Gaussian TCMs developed in this research were shown to build sharper temporal models than comparable TRBMs because they can adaptively select their dependence on the past during training. In the future, similar networks may be used to improve performance in many practical tasks requiring automated analysis of temporal data.

References

[1] Rabiner, L. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE 77* **2**: 257-286.

[2] Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In Parallel Distributed Processing, chapter 8. Cambridge, MA: MIT Press.

[3] Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation* **9**:1735-1780. Cambridge, MA: MIT Press.

[4] Jaeger, H., Lukoševičius, M., Popovici, D., & Siewert, U.. (2007). Optimization and Applications of Echo State Networks with Leaky-Integrator Neurons. *Neural Networks* **20**(3):335-352.

[5] Hurri, J. & Hyvärinen, A. (2002). A Novel Temporal Generative Model of Natural Video as an Internal Model in Early Vision. In *Proceedings of the First International Workshop on Generative-Model-Based Vision* (GMBV 2002). 33-38.

[6] Hinton, G. (2007). To Recognize Shapes, First Learn To Generate Images. *Progress in Brain Research* **165** :535-547

[7] Sutskever, I., & Hinton, G. (2007). Learning Multilevel Distributed Representations for High-Dimensional Sequences. AISTATS 2007.

[8] Ackley, D., Hinton, G., Sejnowski, T. (1985). A Learning Algorithm for Boltzmann Machines. *Cognitive Science* **9**:147-169.

[9] Hinton, G., Dayan, P., To, A., & Neal, R. (1995) The Helmholtz Machine Through Time. In F. Fogelman-Soulie and R. Gallinari, eds. *International Conference on Artificial Neural Networks* (ICANN-95). 483-490.

[10] Hinton, G., Osindero, S., & Teh, Y. (2006) A Fast Learning Algorithm For Deep Belief Networks. *Neural Computation* **18** (7):1527-1554.

[11] Hinton, G. (2002). Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation* **14** (8):1771-1800.

[12] Sutskever, I., Hinton, G., & Taylor, G. (2008). The Recurrent Temporal Restricted Boltzmann Machine, NIPS*21, 2008.

[13] Jaeger, H. (2007). Discovering Multiscale Dynamical Features with Heirarchical Echo State Networks. Tech Report 10, July 2007. College of Engineering and Science, Jacobs University, Denmark.