Sequence Classification Restricted Boltzmann Machines With Gated Units

Son N. Tran¹, Artur d'Avila Garcez², Tillman Weyde, Jie Yin³, Qing Zhang, and Mohan Karunanithi

Abstract—For the classification of sequential data, dynamic Bayesian networks and recurrent neural networks (RNNs) are the preferred models. While the former can explicitly model the temporal dependencies between the variables, and the latter have the capability of learning representations. The recurrent temporal restricted Boltzmann machine (RTRBM) is a model that combines these two features. However, learning and inference in RTRBMs can be difficult because of the exponential nature of its gradient computations when maximizing log likelihoods. In this article, first, we address this intractability by optimizing a conditional rather than a joint probability distribution when performing sequence classification. This results in the “sequence classification restricted Boltzmann machine” (SCRBM). Second, we introduce gated SCRBM (gSCRBM), which use an information processing gate, as an integration of SCRBM with long short-term memory (LSTM) models. In the experiments reported in this article, we evaluate the proposed models on optical character recognition, chunking, and multiresident activity recognition in smart homes. The experimental results show that gSCRBM achieve the performance comparable to that of the state of the art in all three tasks. gSCRBM require far fewer parameters in comparison with other recurrent networks with memory gates, in particular, LSTMs and gated recurrent units (GRUs).

Index Terms—Recurrent neural networks (RNNs), restricted Boltzmann machines, sequence classification, temporal learning.

I. INTRODUCTION

MODELING sequences is an important research topic with a variety of applications, ranging from natural language processing [1], [2] to computer vision [3]. While some studies focus on predicting time-series events [4], [5], classification with sequential data also receives significant attention [6]–[8]. A sequence can be associated with one label for a full input sequence or with a sequence of labels, typically one for each element of the sequence. In [10] and [11], the terms conventional and strong are established, respectively,

for these classifications for tasks. In this article, for ease of presentation, we use the term sequence classification to refer to the strong case, i.e., the labeling of each element of the sequence. Solutions to this classification problem can enable a wide range of real-world applications. For example, speech recognition transcribes a sequence of acoustic feature vectors with spoken words, optical character recognition (OCR) converts images of handwritten or printed text into machine-encoded text, and activity recognition predicts human actions from a sequence of sensor data.

The sequence classification problem has attracted research in dynamic Bayesian models, such as hidden Markov models (HMMs) [11] and conditional random fields (CRFs) [12]. An advantage of these models is the ability to learn the relationships between sequence labels, which is useful for temporal reasoning. However, recent research has seen an increasing interest in recurrent neural networks (RNNs) for sequence classification. Different from dynamic Bayesian models, most RNN models assume that the class labels in a sequence are independent, given the sequence inputs. This makes inference easier but sacrifices dynamic inference based on the temporal dependencies between the sequence labels. A key advantage of RNNs is the ability to learn the temporal representations from data using recurrent hidden layers; however, they have a problem of vanishing/exploding gradients, especially when learning from long sequences using backpropagation through time [13]. This issue can be addressed by incorporating different memory gates in hidden layers, as shown in long short-term memory (LSTM) [13] and gated recurrent units (GRUs) [14]. There have been many attempts to combine the advantages of representation learning with dynamic inference. Most approaches integrate a dynamic Bayesian model and deep neural network [15], e.g., by placing a CRF on top of a bidirectional LSTM (bi-LSTM) [16]. Another approach is the recurrent temporal restricted Boltzmann machine (RTRBM), an extension of the restricted Boltzmann machine (RBM). The RTRBM is a generative graphical model that represents a distribution of sequences and has the ability to learn hidden features [5], [17]. However, learning and inference in RTRBMs are difficult because of the high complexity of computing a joint distribution.

A. Contributions

As the first contribution of this article, we propose a novel and compact model based on RBMs, which we call sequence classification RBMs (SCRBM), to support representation learning and dynamic inference on the classification

Manuscript received July 30, 2018; revised March 26, 2019 and September 30, 2019; accepted December 1, 2019. (Corresponding author: Son N. Tran.)

S. N. Tran is with the Discipline of Information and Communication Technology, University of Tasmania, Hobart, TAS 7005, Australia (e-mail: sn.tran@utas.edu.au).

A. Garcez and T. Weyde are with the Department of Computer Science, City, University of London, London EC1V 0HB, U.K. (e-mail: a.garcez@city.ac.uk; t.e.veyde@city.ac.uk).

J. Yin is with the Discipline of Business Analytics, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: jie.yin@sydney.edu.au).

Q. Zhang and M. Karunanithi are with the Australian E-Health Research Centre, CSIRO, Brisbane, QLD 4209, Australia (e-mail: qing.zhang@csiro.au; mohan.karunanithi@csiro.au).

Color versions of one or more of the figures in this article are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TNNLS.2019.2958103

2162-237X © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

of sequences. The SCRBM is constructed by rolling RBMs with their class nodes over time. Each RBM at time t has a layer of visible units (X^t) and a layer of hidden units (H^t). Together with class nodes Y^t denoting the labels at time t , they form a model representing a distribution: $p(y^{1:T}, \mathbf{x}^{1:T}, \mathbf{h}^{1:T})$. When it comes to inference, there are two questions to answer. First, in order to compute gradients, one needs to infer the hidden states, given the state of the input layer and the class labels from the distribution $p(H^{1:T} | \mathbf{x}^{1:T}, y^{1:T})$. This can be done using variational methods, i.e., treating a hidden unit as a mean field, similar to [5]. Second, to predict the state of the class labels given the state of the input layer, in the best case, one can search for the most probable labels from the conditional distribution, i.e., solving $\arg \max_{Y^{1:T}} p(Y^{1:T} | \mathbf{x}^{1:T})$. In this article, we show that this type of inference can be carried out efficiently by propagating the expectation of the prediction for the previous labels to compute the state of the hidden layer. In other words, the hidden state at time t is dependent on the previous prediction (class labels at $t - 1$), and thus, by association, the state of the class labels at any time point is dependent on the prediction of the labels at the previous time point. For learning, we maximize the log likelihood of the training data. It has been shown that learning a local RBM with labels is tractable [18], [19], but learning the entire sequence is not. This is because it is not easy to marginalize out hidden units in the sequential case, which can be done analytically in the case of a single discriminative RBM. Also, computation becomes expensive due to the exponential growth of possible assignments to the sequence of labels. To solve this problem, we use the above-mentioned mean-field technique to factorize the conditional probability of a sequence into a product of probabilities of local discriminative RBMs.

One drawback of SCRBM is that it cannot capture long-term information and is prone to the problem of vanishing/exploding gradient similar to RNNs [13]. Therefore, in the second contribution of this article, we improve the performance of SCRBM by proposing an idea to integrate SCRBM with LSTM through an information processing gate. This model is called gated SCRBM (gSCRBM). The idea is to take advantage of LSTM cells that use different types of memory gates to handle temporal information. In particular, besides the hidden state, an LSTM cell maintains a cell state to convey the information along the chain over time, while memory gates are used to decide which information should be added or removed from the cell state. In gSCRBM, the cell state is the same as in LSTMs and we integrate a memory gate layer into the hidden layer of the SCRBM. By doing this, we keep the information processing mechanism as it is in LSTMs while allowing probabilistic inference of the class layer as done in SCRBM.

In our experiments, we evaluate the SCRBM and gSCRBM models on three tasks: OCR, Conll 2000 chunking, and multiresident activity recognition in a smart home. The results show that gSCRBM outperforms the state of the art in all three tasks.

Despite having a simpler structure, SCRBM achieves the promising results in OCR and, notably, the highest accuracy for chunking. This motivated further empirical exploration to

compare SCRBM with RNNs, especially with ones having complex memory gates such as GRUs [14] and LSTMs [13]. The results show that the performance of the SCRBM, in some cases, is comparable to that of GRUs and LSTMs, even though SCRBM is considerably more compact, requiring far fewer parameters. The source code for the proposed models can be found at <https://github.com/sontrani/scrblm/>.

The remainder of this article is organized as follows. In Section II, we discuss the related literature. Section III presents the SCRBM model. In Section IV, we propose the integration of LSTM gating techniques into the SCRBM, leading to the gSCRBM. Section V describes the empirical evaluation of the proposed models. Section VI concludes this article and discusses the future work.

II. RELATED WORK

A recent work on dynamic Bayesian models has focused on improving the learning of CRFs by using gradient boosting techniques, such as second-order gradient boosting [8] and gradient tree boosting [20]. In order to incorporate representation learning into CRFs, Do and Artieres [15] proposed neural CRF, which extends CRFs by using neural networks to represent energy functions. On the side of neural networks, LSTMs have been the dominant approach as they can mitigate the problem of vanishing gradients. Other variants of gated neural network are also used in a wide range of sequence classification problems, which can be categorized into architecture variants and cell variants [21]. In terms of network architectures, bi-LSTM where two LSTMs are coupled together, one for forward inference and another for backward inference, has been successful in language processing [22]. In [16], a CRF is placed on top of a bi-LSTM, in which the lower part is used for representation learning and the upper part is used for dynamic inference. In terms of cells, GRU [14] is another variant of gated RNNs, which reduces the complexity compared to LSTMs by combining input and forget gates and sharing values between the cell state and hidden states. Another variant of LSTM is peephole LSTM where cell states are added to the gates [23].

Besides the Bayesian and neural approaches discussed earlier for sequence classification, modeling sequence data with RBMs has been studied previously [4], [5]. However, as generative models, they are not easy to apply to classification tasks. The key problem is that the exact gradient cannot be computed analytically so that approximation algorithms have to be used. By contrast, SCRBM is a discriminative model whose log likelihood is tractable, building on the work on discriminative RBMs. In addition to tractability, another motivation for having a discriminatively learned variant of RTRBMs is the desire for better classification performance. It has been shown in [24] that with sufficient training examples, discriminative learning tends to do better on the task it is optimized for than its generative counterpart model. This has been confirmed by the empirical results of classRBM on nonsequential data [18]. Differently from that work, SCRBM as proposed here is designed for classification with sequence data. Graphically, an SCRBM can be seen as a classRBM with recurrent connections between hidden units, similar to

the relation between RNNs and feedforward neural networks. However, the design of the SCRBM is less straightforward as we have to solve the issue of intractability, as detailed in Section III.

In recent work, the recurrent temporal discriminative RBM (RTD-RBM) has been proposed [25]. It is a generalized version of the RTRBM tailored for discriminative inference. In the SCRBM, we use a similar approach to the RTD-RBM but with a different architecture and a broad evaluation (the RTD-RBM was only evaluated on melody prediction). Also, RTD-RBMs require more parameters than SCRBM because RTD-RBMs generalize RTRBMs by including connections from the previous hidden layers to the current class labels. Another related model is the dynamic Boltzmann machine [26] that supports online learning for sequence prediction, mainly applied to regression. Combining LSTM and RTRBM has been studied in [27], but for generation rather than classification.

III. SCRBM

A. Model

An SCRBM is constructed by rolling RBMs with class labels over time. The model defines a probability distribution

$$p(y^{1:T}, \mathbf{x}^{1:T}, \mathbf{h}^{1:T}) = \prod_{t=1}^T p(y^t, \mathbf{x}^t, \mathbf{h}^t | \mathbf{h}^{t-1}) \quad (1)$$

where $\mathbf{x}^{1:T}$, $\mathbf{h}^{1:T}$ are time series of the visible and hidden states, $y^{1:T}$ is the class-label sequence, and $\mathbf{h}^0$ are the biases of the hidden units.

The main problem of this model, as highlighted in [4], is that inference is intractable. This, however, can be solved by adding recurrent connections, as done for the RTRBM [5]. In RTRBM, class labels are not included. In SCRBM, the local distribution at time t , $p(y^t, \mathbf{x}^t, \mathbf{h}^t | \mathbf{h}^{t-1})$, is replaced by

$$p(y^t, \mathbf{x}^t, \mathbf{h}^t | \hat{\mathbf{h}}^{t-1}) = \frac{\exp(-E_\theta(y^t, \mathbf{x}^t, \mathbf{h}^t; \hat{\mathbf{h}}^{t-1}))}{\sum_{y', \mathbf{x}', \mathbf{h}'} \exp(-E_\theta(y', \mathbf{x}', \mathbf{h}'; \hat{\mathbf{h}}^{t-1}))} \quad (2)$$

where $\hat{\mathbf{h}}^{t-1}$ is the vector of expected values of the hidden units at $t-1$

$$\hat{\mathbf{h}}^{t-1} = \mathbb{E}[\mathbf{H}^{t-1} | \mathbf{x}^{1:t-1}, y^{1:t-1}] \quad (3)$$

with the local energy function

$$E_\theta(y^t, \mathbf{x}^t, \mathbf{h}^t; \hat{\mathbf{h}}^{t-1}) = -[(\mathbf{x}^t)^\top \mathbf{W}_{xh} + \mathbf{u}_{y^t}^\top + (\hat{\mathbf{h}}^{t-1})^\top \mathbf{W}_{hh}] \mathbf{h}^t - \mathbf{a}^\top \mathbf{x}^t - b_{y^t} - \mathbf{c}^\top \mathbf{h}^t \quad (4)$$

which is characterized by the parameters: $\theta = \{\mathbf{W}_{xh}, \mathbf{W}_{hh}, \mathbf{U}, \mathbf{a}, \mathbf{b}, \mathbf{c}\}$. The local energy function (4) is an extension of the standard notation of the energy function of RBMs [28]. The total energy of an SCRBM is the sum of all local energy functions from time 1 to T . It represents the correlation between the hidden units and the external units (input, labels, and previous hidden) with weight matrices applied, and it is also used to compute a probability distribution, as shown in (2). Here, $\mathbf{W}_{xh}$ is the weight matrix between visible units

Fig. 1. Graphical representation of (a) RNN cell and (b) SCRBM cell. The green arrows indicate the directed connections from the previous hidden nodes; the red and blue arrows (resp. lines) indicate the directed (resp. undirected) connections from current labels and inputs, respectively.

and hidden units; $\mathbf{W}_{hh}$ is the recurrent/temporal connection weight matrix of the hidden units; $\mathbf{U}$ is the weight matrix between the class units (represented by one-hot vectors) and the hidden units; $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are the biases of the visible units, hidden units, and class units, respectively; and $\mathbf{u}_{y^t}$ is the column vector y^t of $\mathbf{U}$ which is the result of the multiplication of $\mathbf{U}$ and the one-hot vector for y^t .

The set of parameters in (4) can be reduced by omitting the biases of the visible units because it will be canceled out in the conditional distribution calculation. Fig. 1(a) and (b) shows the resulting structure of the SCRBM. It has the same set of parameters as an RNN. However, the SCRBM is very different in terms of its formulation as defined earlier and in terms of inference and learning, as given described in Sections III-B and III-C.

Different from the generative distribution [see (1)], the conditional distribution is tractable. It offers a computational advantage at inference and learning, where sampling is not needed, as shown in Sections III-B and III-C. In a nutshell, with the earlier realization, SCRBM is a tractable RTRBM for sequence classification.

B. Inference

As mentioned earlier, inference for units in the hidden layer given the inputs and labels is easy, as in [5], where each hidden unit can be treated as a mean field as follows:

$$\hat{\mathbf{h}}^t = \sigma(\mathbf{W}_{xh}^\top \mathbf{x}^t + \mathbf{u}_{y^t} + \mathbf{W}_{hh}^\top \hat{\mathbf{h}}^{t-1} + \mathbf{c}^t). \quad (5)$$

For classification with the SCRBM, one would like to search for the most probable assignment of $Y^{1:T}$ given the inputs $x^{1:T}$. In this article, we show that SCRBM can efficiently infer the state of the hidden layer while performing prediction. As discussed earlier, once the class labels at $t-1$ are known, it is straightforward to infer the state of the hidden layer. We now show that the inference of class labels is also easy, given the state of the hidden layer. In particular, from the mean-field values of the hidden layer in the previous time step, we can infer the class labels using the following conditional distribution:

$$p(y^t | \mathbf{x}^t, \hat{\mathbf{h}}^{t-1}) = \frac{\exp(-\mathcal{F}(\mathbf{x}^t, y^t, \hat{\mathbf{h}}^{t-1}))}{\sum_{y'} \exp(-\mathcal{F}(\mathbf{x}^t, y', \hat{\mathbf{h}}^{t-1}))} \quad (6)$$

with free energy

$$\mathcal{F}(\mathbf{x}^t, y, \hat{\mathbf{h}}^t) = -b_y - \sum_j \log(1 + \exp(\mathbf{w}_{xh,j}^\top \mathbf{x}^t + u_{yj} + c_j)) \quad (7)$$

Algorithm 1 Inference With SCRBM

Data: Input: $\mathbf{x}^{1:T}$
Result: Output: $\mathbf{y}^{1:T}$
for $t = 1 : T$ **do**
 set $\hat{\mathbf{y}}^t = p(\mathbf{y}^t | \mathbf{x}^t, \hat{\mathbf{h}}^{t-1})$
 set $y^t = \arg \max_k \hat{y}_k^t$
 set $\hat{\mathbf{h}}^t = \sigma(\mathbf{W}_{xh}\mathbf{x}^t + \mathbf{U}\hat{\mathbf{y}}^t + \mathbf{W}_{hh}\hat{\mathbf{h}}^{t-1} + \mathbf{c}^t)$
end

where $\mathbf{w}_{xh,j}$ is the j th column of the weight matrix $\mathbf{W}_{xh}$ between the hidden units and the units of the visible layer corresponding to the inputs, and u_{yj} is an element of the weight matrix $\mathbf{U}$ between the hidden units and the units $\mathbf{Y}$ corresponding to the class labels. Here, the visible biases $\mathbf{a}$ have been canceled out, which makes the number of parameters equivalent to that of a standard RNN with the same number of hidden units, i.e., $\theta = \{\mathbf{W}_{xh}, \mathbf{W}_{hh}, \mathbf{U}, \mathbf{b}, \mathbf{c}\}$.

As opposed to the joint distribution in (2), the conditional distribution (6) is tractable, i.e., it can be computed analytically. The difference lies in the denominators of the two distributions: $\sum_{\mathbf{y}', \mathbf{x}', \mathbf{h}'} \exp(-E_\theta(\mathbf{y}', \mathbf{x}', \mathbf{h}'; \hat{\mathbf{h}}^{t-1}))$ of the joint distribution and $\sum_{\mathbf{y}'} \exp(-\mathcal{F}(\mathbf{x}^t, \mathbf{y}^t, \hat{\mathbf{h}}^{t-1}))$ of the conditional distribution. While the joint distribution sums over all possible values of the input, label, and hidden states, the conditional distribution only needs to perform a summation over all possible values of labels, which is much more feasible. This also helps to simplify the learning with SCRBM as we will show in Section III-C.

From (6), we can predict the value of the class labels. Once $\mathbf{y}^t$ is known, we use it to infer the mean-field values $\hat{\mathbf{h}}^t$ as in (5). Let us put this in a specific context, starting from $t = 1$, the conditional distribution $p(y^1 | \mathbf{x}^1, \hat{\mathbf{h}}^0)$ can be computed exactly by marginalizing out the hidden variable $\mathbf{h}^1$ while having $\hat{\mathbf{h}}^0$ as parameters. In order to calculate the values of the current step from the prediction of the previous step, we do not use its predicted value of y^1 , instead we use the distribution to infer the mean-field value $\hat{\mathbf{h}}^1$. This value is then passed to the next prediction step and so on. The details of the SCRBM inference algorithm are given in Algorithm 1.

From Algorithm 1, we can see that SCRBM can capture the dependences between class variables over time through the inference of hidden units using the expected values of the class units.

C. Learning

In SCRBM, we are interested in learning the conditional distribution

$$p(\mathbf{y}^{1:T} | \mathbf{x}^{1:T}) = \frac{p(\mathbf{x}^{1:T}, \mathbf{y}^{1:T})}{\sum_{\mathbf{y}^{1:T}} p(\mathbf{x}^{1:T}, \mathbf{y}^{1:T})} \quad (8)$$

However, it is difficult to marginalize out all hidden variables $\mathbf{h}^{1:T}$ to compute this distribution exactly. The complexity of learning our model would increase exponentially with the length of the sequence, due to the need to sum over all possible combinations of classes at every time step. Thus, instead of computing the distribution directly, we simplify it

by marginalizing out the hidden variable at each time t using the expectation of the hidden state at the previous time $t - 1$. Let us consider

$$\begin{aligned} p(\mathbf{x}^{1:T}, \mathbf{y}^{1:T}) &= \sum_{\mathbf{h}^{1:T}} p(\mathbf{y}^{1:T}, \mathbf{x}^{1:T}, \mathbf{h}^{1:T}) \\ &= \sum_{\mathbf{h}^{1:T}} \prod_{t=1}^T p(\mathbf{y}^t, \mathbf{x}^t, \mathbf{h}^t | \mathbf{h}^{t-1}). \end{aligned} \quad (9)$$

If we first compute the expectation of $\mathbf{h}^{t-1}$ given the previous input states $\mathbf{x}^{1:t-1}$ and $\mathbf{y}^{1:t-1}$, which is equivalent to minimizing the total energy function of the SCRBM, then we have

$$q(\mathbf{x}^{1:T}, \mathbf{y}^{1:T}) = \prod_{t=1}^T p(\mathbf{y}^t, \mathbf{x}^t | \hat{\mathbf{h}}^{t-1}). \quad (10)$$

One can view this as an expectation step to be followed by an optimization step, which maximizes the log likelihood of this simplified distribution

$$\begin{aligned} q(\mathbf{y}^{1:T} | \mathbf{x}^{1:T}) &= \frac{\prod_{t=1}^T p(\mathbf{y}^t, \mathbf{x}^t | \hat{\mathbf{h}}^{t-1})}{\sum_{\mathbf{y}^{1:T}} \prod_{t=1}^T p(\mathbf{y}^t, \mathbf{x}^t | \hat{\mathbf{h}}^{t-1})} \\ &= \frac{\prod_{t=1}^T p(\mathbf{y}^t, \mathbf{x}^t | \hat{\mathbf{h}}^{t-1})}{\prod_{t=1}^T \sum_{\mathbf{y}^t} p(\mathbf{y}^t, \mathbf{x}^t | \hat{\mathbf{h}}^{t-1})} \\ &= \prod_{t=1}^T p(\mathbf{y}^t | \mathbf{x}^t, \hat{\mathbf{h}}^{t-1}). \end{aligned} \quad (11)$$

Since $p(\mathbf{y}^t | \mathbf{x}^t, \hat{\mathbf{h}}^{t-1})$ is tractable as shown in (6), we can compute the above-mentioned distribution exactly. Now, one can learn the model by maximizing the log-likelihood function

$$\ell = \sum_{\mathbf{y}^{1:T}, \mathbf{x}^{1:T}} \sum_{t=1}^T \log p(\mathbf{y}^t | \mathbf{x}^t, \hat{\mathbf{h}}^{t-1}). \quad (12)$$

Similar to other time-series connectionist models, such as standard RNNs and RTRBMs, we train the model using backpropagation through time. The update of the model's set of parameters, denoted by θ , is shown in the following:

$$\nabla \theta = \sum_{t=1}^T \left(\frac{\partial \theta \log p(\mathbf{y}^t | \mathbf{x}^t, \hat{\mathbf{h}}^{t-1})}{\partial \theta} + \frac{\partial \theta \hat{\mathbf{h}}^t}{\partial \theta} \mathcal{O}^t \right) \quad (13)$$

where $(\partial \theta \hat{\mathbf{h}}^t) / \partial \theta = (\partial \theta \sigma(\mathbf{W}_{xh}^\top \mathbf{x}^t + \mathbf{u}_{y^t} + \mathbf{W}_{hh}^\top \hat{\mathbf{h}}^{t-1} + \mathbf{c}^t)) / \partial \theta$ and $(\partial \theta \log p(\mathbf{y}^t | \mathbf{x}^t, \hat{\mathbf{h}}^{t-1})) / \partial \theta$ are local derivatives, and $\hat{\mathbf{h}}^{t-1}$ is a value, not a function of θ ; and for mathematical convenience

$$\mathcal{O}^t = \mathbf{W}_{hh}^\top \hat{\mathbf{h}}^{t+1} (1 - \hat{\mathbf{h}}^{t+1}) \mathcal{O}^{t+1} + \frac{\partial \log p(\mathbf{y}^{t+1} | \mathbf{x}^{t+1}, \hat{\mathbf{h}}^t)}{\partial \hat{\mathbf{h}}^t}. \quad (14)$$

IV. GSCRBM

Section III showed how SCRBM is constructed, learned, and used for the classification of sequences. However, similar to RNNs, SCRBM is not able to model long-term dependences and is prone to the problem of vanishing/exploding gradients [13]. In order to address these points, in this section, we show how to integrate memory gates such as

Fig. 2. (a) LSTM cell. (b) SCRBM cell with gated output.

in LSTM with SCRBM, which results in the gSCRBM. A graphical representation of hidden layers in gSCRBM is shown in Fig. 2(b). It is similar to the LSTM in Fig. 2(a), except that we take one memory gate (the output gate in this case) and integrate it into the SCRBM.

The information processing in gSCRBM operates as follows:

$$\mathbf{i}^t = \mathbf{W}_{xi}\mathbf{x}^t + \mathbf{W}_{hi}\mathbf{h}^{t-1} + \mathbf{b}_i \quad (15)$$

$$\mathbf{f}^t = \mathbf{W}_{xf}\mathbf{x}^t + \mathbf{W}_{hf}\mathbf{h}^{t-1} + \mathbf{b}_f \quad (16)$$

$$\mathbf{o}^t = \mathbf{W}_{xo}\mathbf{x}^t + \mathbf{W}_{ho}\mathbf{h}^{t-1} + \mathbf{b}_o \quad (17)$$

$$\tilde{\mathbf{c}}^t = \mathbf{W}_{xc}\mathbf{x}^t + \mathbf{W}_{hc}\mathbf{h}^{t-1} + \mathbf{b}_c \quad (18)$$

$$\hat{y}_k^t = \frac{\exp(b_k) \prod_j (1 + \exp(o_j^t + u_{kj}))}{\sum_{k'} \exp(b_{k'}) \prod_j (1 + \exp(o_j^t + u_{k'j}))} \quad (19)$$

$$y^t = \arg \max_k \hat{y}_k^t \quad (20)$$

$$\hat{\mathbf{o}}^t = \mathbf{o}^t + \mathbf{U}\hat{y}^t \quad (21)$$

$$\mathbf{c}^t = \sigma(\mathbf{f}^t) * \mathbf{c}^{t-1} + \sigma(\hat{\mathbf{i}}^t) * \tilde{\mathbf{c}}^t \quad (22)$$

$$\mathbf{h}^t = \sigma(\hat{\mathbf{o}}^t) * \tanh(\mathbf{c}^t). \quad (23)$$

The computations of the input gate, forget gate, output gate, and $\tilde{\mathbf{c}}$, which can be seen in (15)–(18), respectively, are similar to the LSTM. In order to integrate the output gate into the SCRBM, we add a mean-field unit ($\hat{\mathbf{o}}$) for that gate, similar to the approach in Section III-A. The same can be done for other gates in LSTMs; however, in our experiments, we found that a gSCRBM with input gate only (denoted as gSCRBM_i) and a gSCRBM with output gate only (denoted as gSCRBM_o) perform better than gSCRBM with more gates. This indicates that the input and output gates contain the most predictive information.

For the input gate, we replace (19) and (21)–(23) by the following equations, respectively:

$$\hat{y}_k^t = \frac{\exp(b_k) \prod_j (1 + \exp(i_j^t + u_{kj}))}{\sum_{k'} \exp(b_{k'}) \prod_j (1 + \exp(i_j^t + u_{k'j}))} \quad (24)$$

$$\hat{\mathbf{i}}^t = \mathbf{i}^t + \mathbf{U}\hat{y}^t \quad (25)$$

$$\mathbf{c}^t = \sigma(\mathbf{f}^t) * \mathbf{c}^{t-1} + \sigma(\hat{\mathbf{i}}^t) * \tilde{\mathbf{c}}^t \quad (26)$$

$$\mathbf{h}^t = \sigma(\mathbf{o}^t) * \tanh(\mathbf{c}^t). \quad (27)$$

In this case, the cell state is updated with the probability estimation of labels through input gate, as shown in (26).

V. EXPERIMENTS

A. OCR

1) *Data Set*: The MIT OCR data set¹ is a widely used benchmark for evaluating sequence classification

algorithms [29]. We use two popular partitions from [7] and from [9] and [16]. In the former, called here “ms” for model selection, the data are partitioned into ten groups, each consisting of training, validation, and test sets. We select the models based on the performance on the validation set and report their average accuracy on the test sets. In the latter, here called “cv” for cross validation, the data are divided into ten folds without validation sets for model selection.

2) *Evaluation Method*: Each model is expected to predict the correct label corresponding to the image of a character as it is drawn. All the models are evaluated using the average classification accuracy per sequence $E(y, y^*)$, where y and y^* are the predicted and the true sequence sets, respectively, as follows:

$$E(y, y^*) = \frac{1}{N} \sum_{i=1}^N \left[\frac{1}{L_i} \sum_{j=1}^{L_i} \mathcal{I}((y_i)_j \neq (y_i^*)_j) \right] \quad (28)$$

where N is the total number of test examples, L_i is the length of the i th sequence, and $\mathcal{I}$ is the 0 – 1 indicator function.

3) *Model Comparison*: We compare the performance of SCRBM on the above-mentioned sequence labeling task with the following models: multiclass support vector machines (SVM_{multiclass}) [30], structured support vector machines (SVM_{struct}) [31], max-margin Markov network (M3N) [29], averaged perceptron [32], search-based structured prediction (SEARN) [33], CRF [12], [34], HMM [11], LogitBoost [35], TreeCRF [20], RTDRBM [25] (using the inference algorithm proposed in this article to adapt to the sequence labeling task), and state-of-the-art models:

- 1) *Structured Learning Ensemble (SLE)* [7]: An optimized ensemble of 7 models: SVM_{multiclass}, SVM_{struct}, M3N, Perceptron, SEARN, CRF, and HMM.
- 2) *Neural CRF* [15]: A combination of CRF and deep networks.
- 3) *Gradient Boosting CRF (GBCRF)* [8]: CRF trained by a gradient boosting algorithm.

4) *Results*: For the “ms” partitioning, a grid search was carried out to determine the best model. We report the best results of the methods evaluated in [7]. For the SCRBM, RTDRBM, and RNNs, the optimized hyperparameters include the learning rate and the number of hidden units. The traditional RNNs employ tanh as the activation function for the hidden units. We also use early stopping, with the performance of the models on the validation set being determined after each epoch. The training was stopped if the validation performance does not improve in ten consecutive epochs. The models have been trained using the Adam algorithm [36]. For the “cv” partition, since the model selection is not possible, we run each model using a different number of hidden units from {50, 100, 500, 1000, 2000, 5000, 10000} and report the lowest average test error rate for the models. The training method is Adam starting at the learning rate of 0.001. In the “ms” partition, this hyperparameter value was found to be an optimal choice for all connectionist models: RNN, GRU, LSTM, SCRBM, and gSCRBM. The number of training epochs is fixed to 30.

Table I shows the individual results. gSCRBM_o outperforms all the other models. It is worth noting that the performance

¹<http://www.seas.upenn.edu/~taskar/ocr/>

TABLE I

AVERAGED TEST SET ERROR RATES (%) OF VARIOUS MODELS ON THE MIT OCR SEQUENCE LABELING TASK

Model	ms	cv
gSCRBM _i	11.414	04.302
gSCRBM _o	11.125	04.178
SCRBM	14.812	04.783
LSTM	11.72	04.76
GRU	14.23	06.38
RTD-RBM	15.46	-
Neural CRF ^{CML}	-	04.44
Neural CRF ^{LM}	-	04.56
SLE	20.58	-
GBCRF	-	04.64
TreeCRF	-	06.99
LogitBoost	-	09.67
RNN	22.92	-
M3N	25.08	13.46
Perceptron	26.40	-
SEARN	27.02	-
SVM _{multiclass}	28.54	-
SVM _{struct}	21.16	-
HMM	23.70	-
CRF	32.30	14.20

of our model is considerably better than that of SLE, neural CRF, and GBCRF, which are the state of the arts to this OCR task.

B. Chunking

The CoNLL 2000 shared task² is a benchmark data set for sequence classification with a focus on chunking. The task is to classify the words in sentences into syntactic parts, e.g., noun phrase (NP) or verb phrase (VP). The data set consists of 8936 and 2012 sentences for training and testing, respectively. We use the binary features from [37] for these data. In this experiment, we use the 50 000 most common features from the training set. The motivation behind the selection of this type of features is that it can help RNNs to achieve better performance than the Glove.6B or word2vec features. Although the word2vec features are smaller in terms of size, generic approaches, such as RNN, GRU, and LSTM, do not perform well, and therefore, more complex variants, i.e., biLSTM [38], bi-LSTM-CRF [16], and CNN-biLSTM-CRF [39], are needed.

Since the data set only includes training and testing samples, we do not perform model selection and early stopping. Instead, we tested different numbers of hidden units [50,100,500,1000] and report the best results on the test set. These values are chosen based on the computational capacity of the machines used for this experiment as a higher number of hidden units would have resulted in computational overload. We use again Adam to take the advantage of the sparsity of the features, the initial learning rate for it is 0.001 as this setting worked very well in the case of OCR mentioned earlier, and also in many other cases from our experience. For evaluation, we use F1 score: $F1 = 2(\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$. The results of the experiments as well as the state-of-the-art results reported in [17] and [41]–[44] are shown in Table II.

²<https://www.clips.uantwerpen.be/conll2000/chunking/>

TABLE II

F1 SCORES FOR THE CONLL 2000 CHUNKING TASK. AGAIN, THE SCRBM

Models	F1 score
gSCRBM _i	95.141
gSCRBM _o	95.050
SCRBM	95.307
LSTM	95.163
GRU	94.719
RNN	94.199
Suzuki et. al. [41]	95.15
Huang et. al. [17]	94.46
Sun et. al. [42]	94.34
Collobert et. al.[43]	94.32
Tsuruoka et. al. [44]	93.81

C. Activity Recognition in Smart Homes

In this experiment, we evaluate our models on activity recognition in smart homes. We use the CASAS data³, which is available from Washington State University and contains data from the smart department testbed with two residents where each resident performing 15 unique activities. The data were collected over 26 days in a smart home equipped with 37 ambient sensors. The data in CASAS are presented in “Date Time Sensor_ID Value Resident_ID Activity” format. For example, “2008-11-10 14:28:17.986759 M22 ON 2 2” shows that resident 2 is hanging up clothes at 14:28:17.986759 on 2008-11-10 when motion sensor M22 is triggered. Similarly, “2008-11-10 14:38:47.974299 M13 OFF 2 8 1 9” means at 14:38:47.974299 on 2008-11-10 when motion sensor M13 is off, and resident 1 is setting the dining room table for dinner, while resident 2 is setting out ingredients for dinner in the kitchen. Different from the previous tasks on OCR and chunking, activity recognition in smart homes is very challenging because of the following reasons. First, the sequences in the data set are considerably long as each of them has been recorded in several hours each day. The minimum length of the sequences is 500 and the maximum length is 866. Second, data collection and annotation are difficult, which results in a small number of samples for training.

We partition the CASAS data into 24 days for training, 1 day for validation, and 1 day for testing. The competitors to our models are the state of the arts used for multiresident activity recognition in smart homes, including factorial HMM [44] and factorial CRF [37], [45]. We also compare with different types of RNNs, denoted as mRNNs, mGRUs, and mLSTMs, which have multiple outputs representing activities of different residents. Since, in this article, we design our models for single output, we combine the activities of multiple residents to a single label. We performed the model selection by using grid search on the number of hidden units and learning rate, similar to the previous experiments on “ms” partition of OCR. For completeness, we carry out a comparison with HMMs, CRFs, RNNs, GRUs, and LSTMs on the same combined labels.

In Tabel III, we show the performance for activities of two residents in the smart house. Each model is tested 20 times and

³<http://ailab.eecs.wsu.edu/casas/>

TABLE III

PREDICTION ACCURACY FOR ALL MODELS ON CASAS DATA SET. R_1 , R_2 , AND R_{all} ARE THE ACCURACY OF PREDICTED ACTIVITIES OF RESIDENT 1, RESIDENT 2, AND THE JOINT ACTIVITIES

	R_1	R_2	R_{all}
gSCRBM _i	92.69	91.46	86.44
gSCRBM _o	90.92	90.78	85.22
SCRBM	81.64	80.06	73.72
GRU	89.84	86.24	81.25
LSTM	91.17	90.47	85.55
mGRU	92.70	88.91	83.10
mLSTM	91.19	89.90	83.23
CRF	76.40	66.07	64.32
RNN	80.03	77.41	71.73
HMM	65.24	65.82	56.58
fHMM	73.55	67.44	55.43
mRNN	78.41	70.07	58.38
fCRF	58.21	56.76	45.84

the averaged prediction accuracy is reported. The performance of a model is measured by the accuracy of each resident's activities and the accuracy of all residents' activities. The former is computed as

$$R_m = \frac{1}{|D_{test}|} \sum_{a^{m,1:T} \in D_{test}} \frac{1}{T} \sum_t (a^{m,t} == \hat{a}^{m,t}) \quad (29)$$

where $a^{m,t}$ and $\hat{a}^{m,t}$ are the ground truth and the predicted activity of resident m at time t , respectively; $a^{m,t} == \hat{a}^{m,t}$ is 1 if the activity of resident m at time t is predicted correctly; otherwise, it is 0. Similarly, the accuracy for activities of all residents is

$$R_{all} = \frac{1}{|D_{test}|} \sum_{a^{1:T} \in D_{test}} \frac{1}{T} \sum_t \left(\bigwedge_m (a^{m,t} == \hat{a}^{m,t}) \right) \quad (30)$$

where $\bigwedge_m$ is the Boolean AND operator and $a^{1:T} \in D_{test}$ is the activities of all residents in the test set.

The results show that by using memory gates, performance of predicting both residents' activities can be improved significantly, e.g., from 58.38% with mRNN to 83.10% and 83.23% with mGRU and mLSTM, respectively, and from 71.73% with RNN to 81.25% and 85.55% with GRU and LSTM, respectively. This is because the lengths of the data sequences are substantial, which bolsters the need for memory gates, while the consistency in daily activities of the residents is high enough to largely reduce the risk of having overfitting. It is shown that SCRBM with input gating achieved the highest performance, better than the other models. In the case of resident 1, multitask GRU has a similar performance with gSCRBM_i but achieved lower accuracy for resident 2 and for joint activities. Note that in the smart house, it is very important to predict exact activities of all residents at a time to understand their collaborative and interactive behaviors.

D. SCRBM Versus GRU and LSTM

As shown in the experiments mentioned earlier, SCRBM has achieved promising empirical results in the OCR and chunking tasks, despite its lack of complex memory gates. This motivates a further investigation into this model in comparison

TABLE IV

AVERAGED TEST SET ERRORS OF RNNs WITH TANH UNITS, GRU, LSTM, AND SCRBM ON THE TAGGING DATA SET

	POS ₅₀₀	POS ₁₀₀₀	POS ₂₀₀₀	POS ₄₀₀₀	POS ₈₀₀₀
RNN	22.921	12.220	10.480	09.207	09.303
GRU	15.853	11.663	09.892	09.011	08.759
LSTM	13.329	12.012	10.168	09.273	08.561
SCRBM	14.808	12.119	10.061	09.178	08.286

with popular variants of RNNs such as gated recurrent units and LSTM, which use complex memory gates in their hidden units, and with a standard RNN using tanh hidden units. It is worth noting that for a visible layer with M units and a hidden layer with N units, the number of weights in the RNN and SCRBM is the same, while GRUs and LSTMs will have, respectively, $2N(M+N+1)$ and $3N(M+N+1)$ more weights than the RNN and SCRBM. In general, if the number of labels is small, then the SCRBM will be approximately three times more compact than the GRU and four times more compact than the LSTM.

First, we evaluate the SCRBM, RNN, GRU, and LSTM on the POS tagging data set from Penn Treebank⁴. The data are partitioned into different training sets with 500, 1000, 2000, 4000, and 2000 examples. Models are selected by holding out 10% of the examples and are then evaluated on a test set with some 1600 sentences. The challenge here is that the lexical features are very large, approximately 450 000. Table IV shows the results where we can see that SCRBM performs competitively in comparison to GRU and LSTM. In particular, SCRBM is better than RNN in all five cases, better than GRU in two cases, and better than LSTM in three cases.

Second, we evaluate the effectiveness of SCRBM, RNN, GRU, and LSTM when they have the same number of parameters: 100 hidden units for both SCRBM and RNN, 33 hidden units for GRU, and 25 hidden units for LSTM. For completeness, we also include in the graphs in the following, GRU and LSTMs with 100 hidden units each. We evaluate the models on the OCR and POS tagging with 2000 training sentences and Conll2000 chunking. Since the original chunking data do not provide a validation set, we used 2.5% of their training data for training and the rest for validation. The original test set was used for testing as provided. In this experiment, for efficiency, we only use three chunking labels as in [37]. The evaluation metrics are the predictive error rate (as before) and the average negative log likelihood on the validation and test sets. All models are trained using Adam with an initial learning rate of 0.001 which, as we found, is generally good for all the models on all three data sets. Figs. 3 and 4 show that SCRBM generalizes better than RNNs, GRU, and LSTM with the same size or number of hidden units. Figs. 3(b.1), (b.2), (c.1), and (c.2) show that in the POS tagging and chunking data sets, all models achieve their best performances very quickly after only a few epochs, with the SCRBM achieving at that point the lowest test set error. Fig. 3(a.1) shows that in the OCR data set, SCRBM performs similar to the LSTM with the same number of 100 hidden units. As for the average negative log likelihood, one can

⁴<http://www.cis.upenn.edu/treebank>

Fig. 3. x -axis: number of epochs. y -axis: predictive errors. All models achieve their best performance very quickly after only a few epochs, with the SCRBM achieving at that point the lowest test set error, except for the OCR data set where the LSTM with 100 hidden neurons is slightly better. (a.1) OCR train. (b.1) POS train. (c.1) Chunking train. (a.1) OCR validation. (b.1) POS 2000 validation. (c.1) Chunking validation. (a.2) OCR test. (b.2) POS 2000 test. (c.2) Chunking test.

Fig. 4. x -axis: number of epochs. y -axis: average negative log likelihood. SCRBM achieves the lowest negative log likelihoods on validation and test sets. (a.1) OCR train. (b.1) POS train. (c.1) Chunking train. (a.1) OCR validation. (b.1) POS 2000 validation. (c.1) Chunking validation. (a.2) OCR test. (b.2) POS 2000 test. (c.2) Chunking test.

see in Figs. 4(a.1), (a.2), (b.1), (b.2), (c.1), and (c.2) that the SCRBM generalizes better than the other models as it achieves the lowest negative log likelihoods on the validation and test sets. This, along with Table IV, explains why in some cases GRUs and LSTMs may be better but only when they have more hidden units.

E. SCRBM Versus BiLSTM and Stacked-LSTM

In the previous experiments, we compare our SCRBM cell with different types of cells, including traditional RNN cell,

TABLE V
gSCRBM_i VERSUS BiLSTM AND STACKEDLSTM

Models	R1	R2	All
gSCRBM _i	92.96	91.46	86.44
BiLSTM	93.15	91.48	86.65
StackedLSTM	93.72	92.11	86.71

GRU cell, and LSTM cell. We also carried out an experiment on LSTM peephole cell on ORC and Chunking data sets, which achieved lower results than our gSCRBM.

In this section, we compare the performance of SCRBM/gSCRBM with advanced LSTM structures [21]. In particular, we choose Bidirectional LSTM (BiLSTM) [38], [46] and Stacked LSTM (StackedLSTM) [47], [48]. Although a direct comparison with (vanilla) SCRBM/gSCRBM may not be fair, experiments in the OCR data set show that gSCRBM_o is slightly better than both BiLSTMs and StackedLSTMs (2 layers), but the differences are not statistically significant. For chunking, BiLSTMs outperform SCRBM but SCRBM is much faster. In Table V, we report the results of SCRBM_i, BiLSTM, and StackedLSTM using the activity recognition data set in Section V-C. It is showed that in this data set, both BiLSTM and StackedLSTM perform slightly better than gSCRBM_i, which is not surprised because gSCRBM_i, as a generic model, does not use bidirectional connections or stacking architecture.

VI. CONCLUSION

We have proposed a simple model for the classification of sequences by rolling RBMs with class labels over time. The main advantages of SCRBM are: it performs representation learning and inference efficiently and it is very compact with a number of parameters equivalent to that of standard RNNs with the same number of hidden neurons. Inference with SCRBM is done by performing prediction of the class and computing mean-field values of the hidden layer at each time t . We train SCRBM by computing the expectation of hidden units at time $t - 1$ for each element of the conditional distribution. This makes learning tractable. More importantly, when coupling SCRBM with complex memory gates in LSTM, we can achieve performance improvement in certain cases. In the experiments, we evaluate the effectiveness of the proposed model in sequence classification on three tasks: OCR, chunking, and multiactivity recognition. In these tasks, we show that the combination of SCRBM and input or output memory gate of LSTM outperforms the state of the art. Furthermore, we show that SCRBM can perform better than all other models in the chunking task, and it can generalize at least as well as GRU and LSTMs without the need for complex memory gates in the hidden units in the case of OCR, POS, and chunking. As future work, we intend to perform further evaluations of the SCRBM at sequence learning. The systematic study of such simple recurrent models, possibly together with the use of knowledge extraction, should offer a better understanding of the basic ingredients for effective sequence learning.

Finally, there has been a resurgence of interest in RBMs although not for sequence classification [49], [50]. In [49],

a regularization technique is proposed to train RBMs for static data using generative methods. In [50], learning of RBMs is accelerated by using parallel computing and hardware design. The addition of regularization, attention, and dropout, as well as parallel speedups, is the possible directions of extensions of SCRBMs.

ACKNOWLEDGMENT

The authors would like to thank S. Cherla for his suggestion and discussion. They would also like to thank C. Sutton for providing the features of the Conll2000 Chunking Task.

REFERENCES

- [1] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proc. 7th conf. Natural Lang. Learn. HLT-NAACL*, Stroudsburg, PA, USA, 2003, pp. 188–191.
- [2] C. Sutton and A. McCallum, "An introduction to conditional random fields," *Found. Trends Mach. Learn.*, vol. 4, no. 4, pp. 267–373, Apr. 2012.
- [3] Y. Cheng, Q. Fan, S. Pankanti, and A. Choudhary, "Temporal sequence modeling for video event detection," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2014, pp. 2235–2242.
- [4] I. Sutskever and G. E. Hinton, "Learning multilevel distributed representations for high-dimensional sequences," in *Proc. Artif. Intell. Statist.*, 2007, pp. 548–555.
- [5] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The Recurrent Temporal Restricted Boltzmann Machine," in *Proc. NIPS*, vol. 2009, pp. 1601–1608.
- [6] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003.
- [7] N. Nguyen and Y. Guo, "Comparisons of sequence labeling algorithms and extensions," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 681–688.
- [8] T. Chen, S. Singh, B. Taskar, and C. Guestrin, "Efficient second-order gradient boosting for conditional random fields," in *Proc. Artif. Intell. Statist.*, 2015, pp. 147–155.
- [9] M. W. Kadous and C. Sammut, "Classification of multivariate time series and structured data using constructive induction," *Mach. Learn.*, vol. 58, nos. 2–3, pp. 179–216, 2005.
- [10] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *ACM SIGKDD Explor. Newslett.*, vol. 12, no. 1, pp. 40–48, 2010.
- [11] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Readings in Speech Recognition*, A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann, 1990, pp. 267–296.
- [12] J. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 282–289.
- [13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
- [14] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.
- [15] T.-M.-T. Do and T. Artieres, "Neural conditional random fields," in *Proc. AISTATS*, 2010, pp. 177–184.
- [16] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," Aug. 2015, *arXiv:1508.01991*. [Online]. Available: <https://arxiv.org/abs/1508.01991>
- [17] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proc. ICML*, 2012, pp. 1881–1888.
- [18] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio, "Learning algorithms for the classification restricted Boltzmann machine," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 643–669, Mar. 2012.
- [19] C. Srikanth, S. N. Tran, A. D. Garcez, and T. Weyde, "Generalising the discriminative restricted Boltzmann machines," in *Proc. ICANN*, 2017, pp. 111–119.
- [20] T. G. Dietterich, G. Hao, and A. Ashenfelder, "Gradient tree boosting for training conditional random fields," *J. Mach. Learn. Res.*, vol. 9, pp. 2113–2139, Oct. 2008.
- [21] Y. Yu, X. Si, C. Hu, and Z. Jianxun, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, 2019.
- [22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
- [23] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proc. IJCNN*, Jul. 2000, pp. 189–194.
- [24] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes," in *Proc. NIPS*, 2002, pp. 841–848.
- [25] S. Cherla, S. N. Tran, A. D. Garcez, and T. Weyde, "Discriminative learning and inference in the Recurrent Temporal RBM for melody modelling," in *Proc. IJCNN*, 2015, pp. 1–8.
- [26] S. Dasgupta and T. Osogami, "Nonlinear dynamic Boltzmann machines for time-series prediction," in *Proc. AAAI*, 2017, pp. 1833–1839.
- [27] Q. Lyu, Z. Wu, J. Zhu, and H. Meng, "Modelling high-dimensional sequences with LSTM-RTRBM: Application to polyphonic music generation," in *Proc. 24th Int. Joint Conf. Artif. Intell. (IJCAI)*, Palo Alto, CA, USA: AAAI Press, 2015, pp. 4138–4139. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2832747.2832827>
- [28] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory" in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA, USA: MIT Press, 1986, pp. 194–281.
- [29] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," *Proc. 16th Int. Conf. Neural Inf. Process. Syst.*, vol. 16, pp. 25–32, 2004.
- [30] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass Kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, Dec. 2001.
- [31] I. Tschantzaris, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," in *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, Sep. 2005.
- [32] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in *Proc. ACL-02 Conf. Empirical Methods Natural Lang. Process.*, Stroudsburg, PA, USA: Association for computational linguistics, 2002, pp. 1–8.
- [33] H. Daumé, Iii, J. Langford, and D. Marcu, "Search-based structured prediction," *Mach. Learn.*, vol. 75, no. 3, pp. 297–325, 2009.
- [34] F. Peng and A. McCallum, "Information extraction from research papers using conditional random fields," *Inf. Process. Manage.*, vol. 42, no. 4, pp. 963–979, 2006.
- [35] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.
- [36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, pp. 1–15, Dec. 2014.
- [37] C. Sutton, A. McCallum, and K. Rohanimanesh, "Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data," *J. Mach. Learn. Res.*, vol. 8, pp. 693–723, May 2007.
- [38] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 4, Jul. 2005, pp. 2047–2052.
- [39] X. Ma and E. Hovy, "End-to-end Sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, Aug. 2016, pp. 1064–1074.
- [40] J. Suzuki and H. Isozaki, "Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data," in *Proc. ACL*, Stroudsburg, PA, USA: The Association for Computer Linguistics, 2008, pp. 665–673.
- [41] X. Sun, L.-P. Morency, D. Okanohara, and J. Tsujii, "Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference," in *Proc. 22nd Int. Conf. Comput. Linguistics COLING*, vol. 1, 2008, pp. 841–848.
- [42] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Nov. 2011.
- [43] Y. Tsuruoka, Y. Miyao, and J. Kazama, "Learning with lookahead: Can history-based models rival globally optimized models?" in *Proc. 15th Conf. Comput. Natural Lang. Learn. CoNLL*, 2011, pp. 238–246.
- [44] T. N. Son, Z. Qing, and K. Mohan, "Improving multi-resident activity recognition for smarter home," in *Proc. IJCAI WS AI IoT*, 2017.
- [45] A. Benmansour, A. Bouchachia, and M. Feham, "Multioccupant activity recognition in pervasive smart home environments," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 34:1–34:36, Feb. 2016.
- [46] B. Yu, Q. Xu, and P. Zhang, "Question classification based on MAC-LSTM," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2018, pp. 69–75.

- [47] S. Fernández, A. Graves, and J. Schmidhuber, "Sequence labelling in structured domains with hierarchical recurrent neural networks," in *Proc. IJCAI*, 2007, pp. 774–779.
- [48] X. Du, H. Zhang, H. Van Nguyen, and Z. Han, "Stacked LSTM deep learning model for traffic prediction in vehicle-to-vehicle communication," in *Proc. IEEE 86th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2017, pp. 1–5.
- [49] D. Chen, J. Lv, and Z. Yi, "Graph regularized restricted Boltzmann machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2651–2659, Jun. 2018.
- [50] L.-W. Kim, "DeepX: Deep learning accelerator for restricted Boltzmann machine artificial neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1441–1453, May 2018.

Son N. Tran received the B.Sc. and M.Sc. degrees in electronic engineering from the Hanoi University of Technology, Hanoi, Vietnam, the EU Erasmus Mundus Joint M.Sc. degree in networks and e-business computing from the University of Reading, Reading, U.K., and the Ph.D. degree in computer science from the City, University of London, London, U.K., in 2016.

He was a Research Fellow with the Commonwealth Scientific and Industrial Research Organisation, Brisbane, QLD, Australia. He is currently a Lecturer with the University of Tasmania, Hobart, TAS, Australia.

Artur d'Avila Garcez received the Ph.D. degree in computer science from Imperial College London, London, U.K.

He is currently the Director of the Research Centre for Machine Learning, City, University of London, London. He has coauthored two books: *Neural-Symbolic Cognitive Reasoning* (Springer, 2009) and *Neural-Symbolic Learning Systems* (Springer, 2002). He has coauthored more than 150 peer-reviewed publications in machine learning, neural computation, and neural-symbolic computing.

Dr. Garcez is also a fellow of the British Computer Society (FBCS). He is also the President of the Neural-Symbolic Learning and Reasoning Association.

Tillman Weyde is currently a Senior Lecturer with the Department of Computer Science, the Group Leader of the Machine Intelligence and Music Informatics Group, a member of the Research Centre for Machine Learning, and the Program Leader of the Data Science Institute, City, University of London, London, U.K. He has published over 100 peer-reviewed articles. His research is focused on processing signals and symbols with machine learning. In particular, he is working on inductive biases that enhance the generalization in deep learning.

Dr. Weyde is also a member of the EPSRC College.

Jie Yin received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology, Hong Kong.

She is currently a Senior Lecturer in business analytics with The University of Sydney, Sydney, NSW, Australia. She has published more than 60 refereed journal and conference papers. Her research interests include data mining, machine learning, and their applications to text mining, network analytics, health informatics, and decision support systems.

Dr. Yin is the Co-Chair of the International Workshop on Social Web for Disaster Management.

Qing Zhang received the B.Sc. degree from Tsinghua University, Beijing, China, and the Ph.D. degree in computer science from the University of New South Wales, Sydney, NSW, Australia.

He is currently a Senior Research Scientist with the Australian e-Health Research Centre, CSIRO, Brisbane, QLD, Australia, and leads the smart home and the Internet-of-Things projects. His research spans sensor data fusion and mining, human identification, and activity recognition.

Dr. Zhang was a Committee Member of the IEEE QLD Section.

Mohan Karunanithi received the Ph.D. degree in biomedical engineering from the University of New South Wales, Sydney, NSW, Australia.

He worked in cardiac research for ten years on ventricular function. He is currently the Group Leader of the Australian e-Health Research Centre (AEHRC), CSIRO, Brisbane, QLD, Australia, directing research projects, scientists, and students in integrated mobile and telehealth health applications in the management of chronic diseases, aged care, and independent living.

Dr. Karunanithi is also a Senior Member of the IEEE Engineering Medicine and Biology Society (EMBS) and the Founder and the Chair of the IEEE EMBS Chapter, QLD, Australia.