# Sequence Classification Restricted Boltzmann Machines with Gated Units

Son N. Tran, Artur d'Avila Garcez, Tillman Weyde, Jie Yin, Qing Zhang, 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 variables, 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 paper, 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 SCRBMs (gSCRBMs), which use of an information processing gate, as an integration of SCRBMs with Long-Short Term Memory (LSTM) models. In the experiments reported in this paper, we evaluate the proposed models on optical character recognition, chunking and multi-resident activity recognition in smart homes. The experimental results show that gSCRBMs achieve performance comparable to that of the state-of-the-art in all three tasks. gSCRBMs 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, Restricted Boltzmann machines, Temporal learning, Sequence classification**

## I. Introduction

MODELLING 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], [6], classification with sequential data also receives significant attention [7], [8], [9]. 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], [11] the terms *conventional* and *strong* are established, respectively, for these classification for tasks. In this paper, for ease of presentation, we use the term *sequence classification* to refer to the *strong* case, i.e. the labelling 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.

Son is with the University of Tasmania, Australia. Qing and Mohan are with the Australian E-Health Research Centre, CSIRO, Australia. Jie Yin is with the University of Sydney. Artur and Tillman are with the City, University of London. Emails: sn.tran@utas.edu.au; {qing.zhang,mohan.karunanithi}@csiro.au; jie.yin@sydney.edu.au; {a.garcez,t.e.weyde}@city.ac.uk.

The sequence classification problem has attracted research in dynamic Bayesian models such as *hidden Markov models* (HMMs) [12] and *conditional random fields* (CRFs) [13]. An advantage of these models is the ability to learn 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 temporal representations from data using recurrent hidden layers, however, they have a problem of vanishing/exploding gradients, especially when learning from long sequences using back-propagation through time [14]. This issue can be addressed by incorporating different memory gates in hidden layers, as shown in *Long Short Term Memory* (LSTM) [14] and *Gated Recurrent Units* (GRUs) [15]. 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 [16], e.g. by placing a CRF on top of a bidirectional LSTM [17]. 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 [6], [18]. 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 paper, we propose a novel and compact model based on restricted Boltzmann machines (RBMs), which we call *sequence classification RBMs* (SCRBM), to support representation learning and dynamic inference on the classification 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, similarly to [6].

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 paper, 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 $t-1$), 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 maximise the log-likelihood of the training data. It has been shown that learning a local RBM with labels is tractable [19], [20], but learning the entire sequence is not. This is because it is not easy to marginalise 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 mean-field technique mentioned above to factorise 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 [14]. Therefore, in the second contribution of this paper we improve the performance of SCRBM by proposing an idea to integrate SCRBM with Long Short Term Memory through an information processing gate. This model is called gated SCRBM (gSCRBM). The idea is to take advantage of LSTM cells which use different types of memory gates to handle temporal information. In particular, besides the hidden state, a 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: optical character recognition (OCR), Conll 2000 chunking, and multi-resident 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 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 [15] and LSTMs [14]. 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/sontranai/scrbm/.

The remainder of the paper is organized as follows. In the next Section II, we discuss the related literature. Section III presents the *sequence classification RBM* (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 paper and discusses future work.

## II. Related work

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 [9] and gradient tree boosting [21]. In order to incorporate representation learning into CRFs, in [16] the authors propose 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 categorised into architecture variants and cell variants [22]. In terms of network architectures, bidirectional LSTMs (bi-LSTM) where two LSTMs are coupled together, one for forward inference and another for backward inference, have been successful in language processing [23]. In [17], 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 [15] is another variant of gated recurrent neural networks, which reduces the complexity compared to LSTMs by combining input and forget gates and sharing values between cell state and hidden state. Another variant of LSTM is peephole LSTM where cell states are added to the gates [24].

Besides the Bayesian and neural approaches discussed above for sequence classification, modelling sequence data with RBMs has been studied previously [5], [6]. However, as generative models, they are not easy to apply to classification tasks. The key problem is that the exact gradient can not be computed analytically, so that of 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 [25] 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 non-sequential data [19]. 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 feed-forward neural networks. However, the design of the SCRBM is less straightforward as we have to solve the issue of intractability, as detailed in the next section.

In recent work, the *Recurrent Temporal Discriminative Restricted Boltzmann Machine* (RTD-RBM) has been proposed [26]. 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


(a) Recurrent neural network cell. (b) SCRBM cell.

Fig. 1: Graphical representation of recurrent neural network and sequence classification restricted Boltzmann machine. The green arrows indicate 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 a broad evaluation (the RTD-RBM was only evaluated on melody prediction). Also, RTD-RBMs require more parameters than SCRBMs because RTD-RBMs generalize RTRBMs by including connections from previous hidden layers to the current class labels. Another related model is the *dynamic Boltzmann machine* [27] which supports online learning for sequence prediction, mainly applied to regression. Combining LSTM and RTRBM has been studied in [28], but for generation rather than classification.

## III. Sequence Classification Restricted Boltzmann Machine (SCRBM)

### A. Model

A *Sequence Classification Restricted Boltzmann Machine* (SCRBM) is constructed by rolling RBMs with class labels over time. The model defines a probability distribution:

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

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; $\mathbf{h}^0$ are the biases of the hidden units.

The main problem of this model, as highlighted in [5], is that inference is intractable. This, however, can be solved by adding recurrent connections, as done for the RTRBM [6]. In RTRBM, class labels are not included. In SCRBMs, the local distribution at time $t$: $p(y^t, \mathbf{x}^t, \mathbf{h}^t | \mathbf{h}^{t-1})$ is replaced by:

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

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

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

with the local energy function:

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

which is characterised 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 [29]. The total energy of an SCRBM is the sum of all local energy functions from time 1 to $T$. It represents the correlation between hidden units and the external units (input, labels, 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 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; $\mathbf{u}_{y^t}$ 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 cancelled out in the conditional distribution calculation. Figures 1a and 1b show the resulting structure of the SCRBM. It has the same set of parameters as a recurrent neural network. However, the SCRBM is very different in terms of its formulation as defined above and in terms of inference and learning as will be described in sub-section III.B and III.C.

Different from the generative distribution (equation 1, the conditional distribution is tractable. It offers a computational advantage at inference and learning, where sampling is not needed, as shown in the next two sections. In a nutshell, with the above 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 [6], where each hidden unit can be treated as a mean-field, as follows:

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

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 paper, we show that SCRBM can efficiently infer the state of the hidden layer while at the same time 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 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:

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

with free energy:

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

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 $U$ between the hidden units and the units Y corresponding to the class labels. Here, the visible biases $\mathbf{a}$ have been cancelled 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: $\sum_{y',\mathbf{x}',\mathbf{h}'} \exp(-E_\theta(y', \mathbf{x}', \mathbf{h}'; \hat{\mathbf{h}}^{t-1}))$ of the joint distribution and $\sum_{y'} \exp\left(-\mathcal{F}(\mathbf{x}^t, y', \hat{\mathbf{h}}^{t-1})\right)$ 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 the next sub-section.

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.

**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 $y^t = \arg\max_k \hat{y}^t_k$
  set $\hat{\mathbf{h}}^t = \sigma(\mathbf{W}_{xh}\mathbf{x}^t + \mathbf{U}\hat{\mathbf{y}}^t + \mathbf{W}_{hh}\mathbf{h}^{t-1} + \mathbf{c}^t)$
**end**

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

### C. Learning

In SCRBMs, we are interested in learning the conditional distribution:

$$p(y^{1:T}|\mathbf{x}^{1:T}) = \frac{p(\mathbf{x}^{1:T}, y^{1:T})}{\sum_{y'^{1:T}} p(\mathbf{x}^{1:T}, y'^{1:T})} \tag{8}$$

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. So, 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:

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

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

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

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

$$\begin{aligned} q(y^{1:T}|\mathbf{x}^{1:T}) &= \frac{\prod_{t=1}^{T} p(y^t, \mathbf{x}^t|\hat{\mathbf{h}}^{t-1})}{\sum_{y'^{1:T}} \prod_{t=1}^{T} p(y'^t, \mathbf{x}^t|\hat{\mathbf{h}}^{t-1})} \\ &= \frac{\prod_{t=1}^{T} p(y^t, \mathbf{x}^t|\hat{\mathbf{h}}^{t-1})}{\prod_{t=1}^{T} \sum_{y'^t} p(y'^t, \mathbf{x}^t|\hat{\mathbf{h}}^{t-1})} \\ &= \prod_{t=1}^{T} p(y^t|\mathbf{x}^t, \hat{\mathbf{h}}^{t-1}) \end{aligned} \tag{11}$$

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

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

Similarly to other time-series connectionist models, such as standard RNNs and RTRBMs, we train the model using back-propagation through time. The update of the model's set of parameters, denoted by $\theta$, is shown below.

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

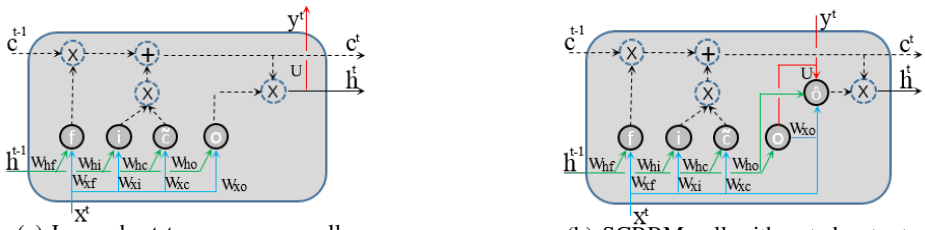
where $\frac{\partial_\theta \hat{\mathbf{h}}^t}{\partial\theta} = \frac{\partial_\theta \sigma(\mathbf{W}_{xh}^\top \mathbf{x}^t + \mathbf{u}_{y^t} + \mathbf{W}_{hh}^\top \mathbf{h}^{t-1} + \mathbf{c}^t)}{\partial\theta}$ and $\frac{\partial_\theta \log p(y^t|\mathbf{x}^t, \mathbf{h}^{t-1})}{\partial\theta}$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

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

## IV. Gated SCRBMs

The previous section showed how SCRBMs are constructed, learned and used for the classification of sequences. However, similar to RNNs, SCRBMs are not able to model long term dependencies and are prone to the problem of vanishing/exploding gradients [14]. In order to address these points, in this section, we show how to integrate memory gates like in *long-short term memory* with SCRBMs, which results in the "gated SCRBM" (gSCRBM). A graphical representation of hidden layers in gSCRBM is shown in Figure 2b. It is similar to the LSTM in Figure 2a 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:




(a) Long short term memory cell. (b) SCRBM cell with gated output.

Fig. 2: Long short term memory and SCRBM with gated units

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

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

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

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

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

$$y^t = \arg\max_k \hat{y}_k^t \tag{20}$$

$$\hat{\mathbf{o}}^t = \mathbf{o}^t + \mathbf{U}\hat{\mathbf{y}}^t \tag{21}$$

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

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

The computations of the input gate, forget gate, output gate and $\tilde{c}$, which can be seen in (15), (16), (17) and (18), respectively, are similar to an 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 gSCRM$_i$), and a gSCRBM with output gate only (denoted as gSCRM$_o$), perform better than gSCRBMs with more gates. This indicates that the input and output gates contain the most predictive information.

For the input gate we replace (19), (21), (22), and (23) by the following equations (24), (25), (26), and (27), respectively:

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

$$\hat{\mathbf{i}}^t = \mathbf{i}^t + \mathbf{U}\hat{\mathbf{y}}^t \tag{25}$$

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

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

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

## V. Experiments

### A. Optical Character Recognition

*1) Dataset:* The MIT OCR dataset[1] is a widely used benchmark for evaluating sequence classification algorithms [30]. We use two popular partitions from [8] and from [16], [9]. In the former, called here 'ms' for model selection, the data is partitioned into 10 groups, each consisting of a training, validation and test set. We select models based on 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 is divided into 10 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:

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

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 sequence labelling task with the following models: Multiclass support vector machines (SVM$_{multiclass}$) [31], Structured support vector machines (SVM$_{struct}$) [32], Max-margin Markov network (M3N)[30], Averaged Perceptron [33], Search-based structured prediction (SEARN) [34], Conditional random field (CRF) [13], [35], Hidden Markov model (HMM) [12], LogitBoost [36], TreeCRF [21], RTDRBM [26] (using the inference algorithm proposed in this paper to adapt to the sequence labelling task), and state-of-the-art models:

- Structured learning ensemble (SLE) [8]: An optimised ensemble of 7 models: SVM$_{multiclass}$, SVM$_{struct}$, M3N, Perceptron, SEARN, CRF and HMM.
- Neural CRF [16]: A combination of CRF and deep networks.
- Gradient boosting CRF (GBCRF) [9]: 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 [8]. For the SCRBMs, RTDRBM, and RNNs, the optimised hyper-parameters 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 10 consecutive epochs. The models have been trained using the *Adam* algorithm [37]. For the 'cv' partition, since 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 learning rate $0.001$. In the 'ms' partition, this hyper-parameter value was found to be an optimal choice for

[1] http://www.seas.upenn.edu/~taskar/ocr/

| 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 |

TABLE I: The averaged test set error rates (%) of various models on the MIT OCR sequence labelling task. The 'ms' variant uses model selection and 'cv' uses cross-validation without model selection.

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 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 task[2] is a benchmark dataset 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 dataset consists of 8936 and 2012 sentences for training and testing, respectively. We use the binary features from [38] for these data. In this experiment, we use the 50000 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[39], bi-LSTM-CRF [17] and CNN-biLSTM-CRF [40] are needed.

Since the dataset 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 above, and also in many other cases from our experience. For evaluation, we use F1 score: $F1 = 2\frac{precision*recall}{precision+recall}$. The results of the experiments as well as the state-of-the-art results reported in [41], [17], [42], [43], [44], are shown in Table II.

| 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 |

TABLE II: F1 scores for the CoNLL 2000 chunking task. Again, the SCRBM with gated input performs better than RNN-based methods. Differently from in the OCR task, combining the SCRBM with output gates achieves lower performance than an LSTM model. In this case, due to the much larger number of features in comparison with the size of the training data, more complex models, such as GRU, LSTM and gSCRBM, seem to overfit while the plain SCRBM, with a much lower number of parameters, achieves the highest accuracy.

### C. Activity Recognition in Smart Homes

In this experiment, we evaluate our models on activity recognition in smart homes. We use the CASAS data[3], 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 was collected over 26 days in a smart home equipped with 37 ambient sensors. The data in CASAS is 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, 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 dataset 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 multi-resident activity recognition in smart homes, including: factorial HMM [45], [46] and factorial CRF [38], [47]. We also compare with different types of recurrent neural networks, denoted as mRNNs, mGRUs, mLSTMs, which have multiple outputs

[2]https://www.clips.uantwerpen.be/conll2000/chunking/

[3]http://ailab.eecs.wsu.edu/casas/

representing activities of different residents. Since in this paper 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.

| | $R_1$ | $R_2$ | $R_{all}$ |
|---|---|---|---|
| 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 |

TABLE III: Prediction accuracy for all models on CASAS dataset. $R_1$, $R_2$, $R_{all}$ are the accuracy of predicted activities of resident 1, resident 2 and the joint activities.

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

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

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:

$$R_{all} = \frac{1}{|D_{test}|} \sum_{\mathbf{a}^{1:T} \in \mathcal{D}_{test}} \frac{1}{T} \sum_t (\bigwedge_m (a^{m,t} == \hat{a}^{m,t})) \qquad (30)$$

where $\bigwedge_m$ is the boolean AND operator and $\mathbf{a}^{1:T} \in \mathcal{D}_{test}$ 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, multi-task GRU has 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 resident at a time to understand their collaborative and interactive behaviors.

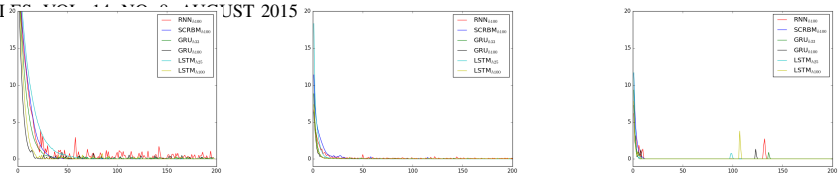
### D. SCRBM versus GRU and LSTM

As shown in the experiments above, 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 with popular variants of recurrent neural networks such as *gated recurrent units* and *long short term memory*, 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 3 times more compact than the GRU and 4 times more compact than the LSTM.

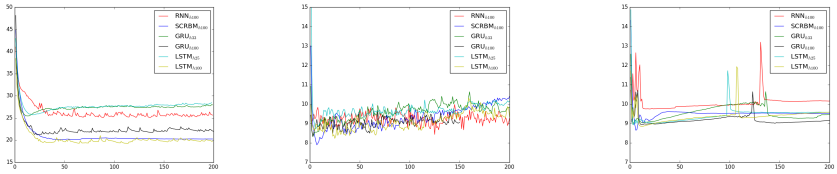
First, we evaluate the SCRBM, RNN, GRU, and LSTM on POS tagging dataset from Penn Treebank[4]. The data is partitioned into different training sets with $500$, $1000$, $2000$, $4000$ and $2000$ examples. Models are selected by holding out $10\%$ of the examples and are then evaluated on a test set with 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 2 cases, and better than LSTM in 3 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 below, GRU and LSTMs with $100$ hidden units each. We evaluate the models on the OCR, POS tagging with $2000$ training sentences, and Conll2000 chunking. Since the original chunking data does 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 3 chunking labels as in [38]. 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 datasets. Figures 3 and 4 show that SCRBM generalises better than RNNs, GRU and LSTM with the same size or number of hidden units. Figures 3b.1, 3b.2, 3c.1 and 3c.2 show that in the POS tagging and chunking datasets, 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. Figure 3a.1 shows that in the OCR dataset, SCRBM performs similarly to the LSTM with the same number of 100 hidden units. As for the average negative log-likelihood, one can see in Figures 4a.1, 4a.2, 4b.1, 4b.2, 4c.1, 4c.2 that the SCRBM generalises 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

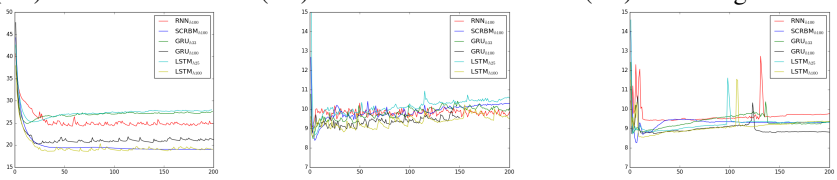
[4]http://www.cis.upenn.edu/treebank



(a.1) OCR train. (b.1) POS train. (c.1) Chunking train.

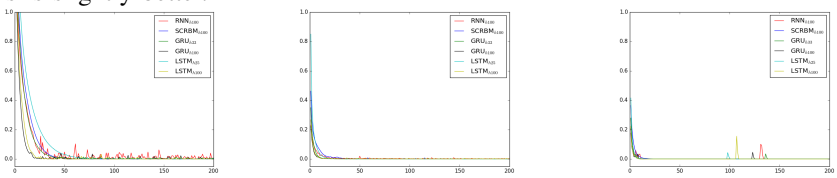


(a.1) OCR validation. (b.1) POS 2000 validation. (c.1) Chunking validation.

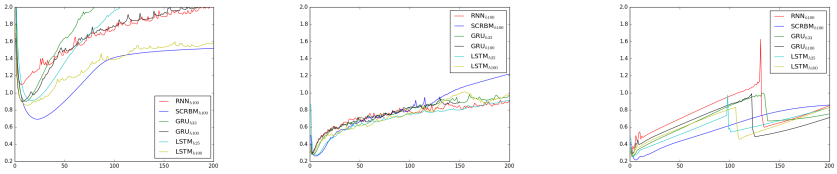


(a.2) OCR test. (b.2) POS 2000 test. (c.2) Chunking test.

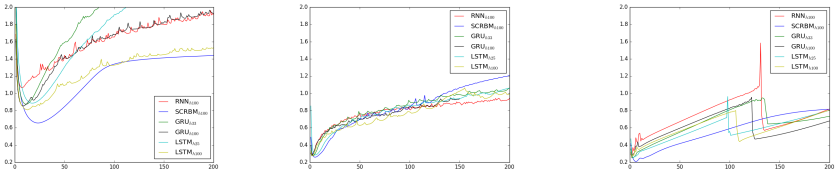
Fig. 3: x-axis: number of epochs; y axis: predictive errors. All models achieve their best performance very quickly after only a few epochs, with the SCRBM achieving at that point the lowest test set error, except for the OCR dataset where the LSTM with 100 hidden neurons is slightly better.



(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.

why in some cases GRUs and LSTMs may be better but only when they have more hidden units.

| | $POS_{500}$ | $POS_{1000}$ | $POS_{2000}$ | $POS_{4000}$ | $POS_{8000}$ |
|---|---|---|---|---|---|
| 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 |

TABLE IV: The averaged test set errors of RNNs with $Tanh$ units, GRU, LSTM and SCRBM on the tagging dataset with different training sets of size 500, 1000, 2000, and the chunking dataset. Model sizes and other hyperparameters were optimised through a grid search. SCRBM achieves performance comparable to LSTMs and GRUs using a much simpler model, e.g. in the OCR dataset, LSTM uses 2000 hidden units while SCRBM uses 100 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, GRU cell, and LSTM cell. We also carried out an experiment on LSTM peephole cell on ORC and Chunking datasets which achieved lower results than our gSCRBM.

In this section, we compare the performance of SCRBM/gSCRBM with advanced LSTM structures [22]. In particular we choose Bi-directional LSTM (BiLSTM) [39], [48] and Stacked LSTM (StackedLSTM) [49], [50]. Although a direct comparison with (vanilla) SCRBM/gSCRBM may not be fair, experiments in the OCR dataset 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 SCRBMs but SCRBMs are

much faster. In Table V we report the results of SCRBM$_i$, BiLSTM and StackedLSTM using the activity recognition dataset in Section V-C. It is showed that in this dataset both BiLSTM and StackedLSTM perform slightly better than gSCRBM$_i$, which is not surprised because gSCRBM$_i$, as a generic model, does not use bi-directional connections or stacking architecture.

| 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 |

TABLE V: gSCRBM$_i$ vs BiLSTM and StackedLSTM

## VI. Conclusion and Future Work

We have proposed a simple model for the classification of sequences by rolling Restricted Boltzmann Machines 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 recurrent neural networks 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 multi-activity 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 generalise 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 SCRBMs 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 [51], [52]. In [51] a regularization technique is proposed to train RBMs for static data using generative methods. In [52] learning of RBMs is accelerated by using parallel computing and hardware design. The addition of regularization, attention, and dropout, as well as parallel speed-ups, are all possible directions of extensions of SCRBMs.

## Acknowledgement

We thank Srikanth Cherla for his suggestion and discussion. We also thank Charles 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 *Proceedings of the Seventh Conference on Natural Language Learning*, Stroudsburg, 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 *Proceedings of CVPR*, 2014, pp. 2235–2242.

[4] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *NIPS*, 2007, pp. 1345–1352.

[5] I. Sutskever and G. E. Hinton, "Learning multilevel distributed representations for high-dimensional sequences." in *AISTATS*, 2007, pp. 548–555.

[6] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The Recurrent Temporal Restricted Boltzmann Machine," in *NIPS*, 2009, pp. 1601–1608.

[7] 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.

[8] N. Nguyen and Y. Guo, "Comparisons of Sequence Labeling Algorithms and Extensions," in *ICML*. ACM, 2007, pp. 681–688.

[9] T. Chen, S. Singh, B. Taskar, and C. Guestrin, "Efficient Second-Order Gradient Boosting for Conditional Random Fields," in *AISTATS*, 2015, pp. 147–155.

[10] M. W. Kadous and C. Sammut, "Classification of multivariate time series and structured data using constructive induction," *Machine learning*, vol. 58, no. 2, pp. 179–216, 2005.

[11] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *SIGKDD Explor. Newsl.*, vol. 12, no. 1, pp. 40–48, Nov. 2010.

[12] L. R. Rabiner, "Readings in speech recognition," A. Waibel and K.-F. Lee, Eds., San Francisco, USA, 1990, ch. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296.

[13] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *ICML*, 2001, pp. 282–289.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[15] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.

[16] T. Do and T. Artieres, "Neural conditional random fields," in *AISTATS*, 2010, pp. 177–184.

[17] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging." *CoRR*, vol. abs/1508.01991, 2015.

[18] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription," in *ICML*, 2012, pp. 1159–1166.

[19] 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.

[20] C. Srikanth, T. Son, N., W. Tillman, and G. Artur, "Generalising the discriminative restricted boltzmann machines," in *ICANN*, 2017.

[21] A. A. Thomas G. Dietterich, Guohua Hao, "Gradient tree boosting for training conditional random fields," *JMLR*, pp. 2113–2139, 2008.

[22] Y. Yu, X. Si, C. Hu, and JianxunZhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural Computation*, 2019.

[23] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[24] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IJCNN.*, vol. 3, July 2000, pp. 189–194 vol.3.

[25] A. Y. Ng and M. I. Jordan, "On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes," in *NIPS*, 2001, pp. 841–848.

[26] S. Cherla, S. N. Tran, A. d. Garcez, and T. Weyde, "Discriminative learning and inference in the recurrent temporal rbm for melody modelling," in *IJCNN*, 2015, pp. 1–8.

[27] S. Dasgupta and T. Osogami, "Nonlinear dynamic boltzmann machines for time-series prediction," in *AAAI*, 2017.

[28] Q. Lyu, Z. Wu, J. Zhu, and H. Meng, "Modelling high-dimensional sequences with lstm-rtrbm: Application to polyphonic music generation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, pp. 4138–4139. [Online]. Available: http://dl.acm.org/citation.cfm?id=2832747.2832827

[29] P. Smolensky, "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1." MIT Press, 1986, ch. Information Processing in Dynamical Systems: Foundations of Harmony Theory, pp. 194–281.

[30] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov Networks," *Advances in neural information processing systems*, vol. 16, p. 25, 2004.

[31] K. Crammer and Y. Singer, "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines," *The Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2002.

[32] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," in *Journal of Machine Learning Research*, 2005, pp. 1453–1484.

[33] M. Collins, "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms," in *ACL Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2002, pp. 1–8.

[34] H. Daumé III, J. Langford, and D. Marcu, "Search-based Structured Prediction," *Machine learning*, vol. 75, no. 3, pp. 297–325, 2009.

[35] F. Peng and A. McCallum, "Information extraction from research papers using conditional random fields," *Information processing & management*, vol. 42, no. 4, pp. 963–979, 2006.

[36] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: a Statistical View of Boosting," *The Annals of Statistics*, vol. 38, no. 2, 2000.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." *CoRR*, vol. abs/1412.6980, 2014.

[38] 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.

[39] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, July 2005, pp. 2047–2052 vol. 4.

[40] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2016, pp. 1064–1074.

[41] J. Suzuki and H. Isozaki, "Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data." in *ACL*. The Association for Computer Linguistics, 2008, pp. 665–673.

[42] X. Sun, L.-P. Morency, D. Okanohara, and J. Tsujii, "Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference," in *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, ser. COLING '08, 2008, pp. 841–848.

[43] 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.

[44] Y. Tsuruoka, Y. Miyao, and J. Kazama, "Learning with lookahead: Can history-based models rival globally optimized models?" in *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, ser. CoNLL '11, 2011, pp. 238–246.

[45] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," *Mach. Learn.*, vol. 29, no. 2-3, pp. 245–273, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1007425814087

[46] T. Son, N., Z. Qing, and K. Mohan, "Improving multi-resident activity recognition for smarter home," in *IJCAI 2017 WS on AI for IoT*, ser. IJCAI'17, 2017.

[47] 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, Dec. 2015.

[48] B. Yu, Q. Xu, and P. Zhang, "Question classification based on mac-lstm," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, June 2018, pp. 69–75.

[49] S. Fernández, A. Graves, and J. Schmidhuber, "Sequence labelling in structured domains with hierarchical recurrent neural networks," in *IJCAI'07*, 2007, pp. 774–779.

[50] X. Du, H. Zhang, H. V. Nguyen, and Z. Han, "Stacked lstm deep learning model for traffic prediction in vehicle-to-vehicle communication," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sep. 2017, pp. 1–5.

[51] D. Chen, J. Lv, and Z. Yi, "Graph regularized restricted boltzmann machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2651–2659, June 2018.

[52] L. Kim, "Deepx: Deep learning accelerator for restricted boltzmann machine artificial neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1441–1453, May 2018.


**Son N. Tran** is a Lecturer at the University of Tasmania, Australia. Son was a research fellow at the Commonwealth Scientific and Industrial Research Organisation, Australia. He holds a Ph.D. in Computer Science (2016) from City, University of London, an EU Erasmus Mundus joint MSc in Networks and e-Business Computing, University of Reading, and a BSc and MSc in Electronic Engineering from the Hanoi University of Technology.

**Artur d'Avila Garcez** , FBCS, is Director of the Research Centre for Machine Learning at City, University of London. He holds a Ph.D. in Computer Science from Imperial College London. He co-authored two books: Neural-Symbolic Cognitive Reasoning (Springer, 2009) and Neural-Symbolic Learning Systems (Springer, 2002), and more than 150 peer-reviewed publications in Machine Learning, Neural Computation and Neural-Symbolic Computing. Garcez is president of the Neural-Symbolic Learning and Reasoning Association.

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

**Jie Yin** received a PhD in Computer Science from the Hong Kong University of Science and Technology. She is a Senior Lecturer in Business Analytics at The University of Sydney, Australia. Her research includes data mining, machine learning, and their applications to text mining, network analytics, health informatics, and decision support systems. She has published more than 60 refereed journal and conference papers. She is a co-chair of the International Workshop on Social Web for Disaster Management.

**Qing Zhang** received a BSc from Tsinghua University, Beijing, China, and a PhD in computer science from University of New South Wales, Sydney, Australia. He is a senior research scientist at the Australian e-Health Research Centre, CSIRO and leads the smart home and internet of things projects. He was committee member of IEEE QLD section. His research spans sensor data fusion and mining, human identification and activity recognition.

**Mohan Karunanithi** obtained his doctorate in Biomedical Engineering at the University of New South Wales. He worked in cardiac research for 10 years on ventricular function. He is Group Leader at the AEHRC, CSIRO, directing research projects, scientists and students in Integrated mobile and telehealth health applications in the management of chronic diseases, aged care and independently living. He is senior member of IEEE Engineering Medicine and Biology Society (EMBS), founder and chair of the IEEE EMBS chapter in Queensland, Australia.