RECURRENT SUPPORT VECTOR MACHINES FOR SPEECH RECOGNITION

Shi-Xiong Zhang, Rui Zhao, Chaojun Liu, Jinyu Li and Yifan Gong

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

{zhashi, ruzhao, chaojunl, jinyli, ygong}@microsoft.com

ABSTRACT

Recurrent Neural Networks (RNNs) using Long-Short Term Memory (LSTM) architecture have demonstrated the state-of-the-art performances on speech recognition. Most of deep RNNs use the softmax activation function in the last layer for classification. This paper illustrates small but consistent advantages of replacing the softmax layer in RNN with Support Vector Machines (SVMs). The parameters of RNNs and SVMs are jointly learned using a sequence-level max-margin criteria, instead of cross-entropy. The resulting model is termed Recurrent SVM. The conventional SVMs need to predefine a feature space and do not have internal states to deal with arbitrary long-term dependencies in sequences. The proposed recurrent SVM uses LSTMs to learn the feature space and to capture temporal dependencies, while using the SVM (in the last layer) for sequence classification. The model is evaluated on the Windows phone task for large vocabulary continuous speech recognition.

Index Terms— Deep learning, LSTM, SVM, maximum margin, sequence training

1. INTRODUCTION

Recurrent Neural Networks (RNNs) [1] and Structured SVMs [2, 3] are two successful models for sequence classification, such as speech recognition [3, 4]. The RNNs are universal models in the sense that they can approximate any sequence-to-sequence mapping to arbitrary accuracy [5, 6]. However, there are three major drawbacks of RNNs. First, the training usually requires to solve a highly nonlinear optimization problem which has many local minima. Second, they tend to overfit given the limited data if training goes on too long [1]. Third, the number of neurons in RNNs is fixed, empirically.

Alternatively, Support Vector Machines (SVMs) supplied the maximum margin classifier idea [7], has attract extensive research interests [8–10]. The SVM was originally proposed for binary classification. It can be extended to handle the multiclass classification or sequence recognition. The modified SVMs for multiclass classification and sequence recognition are known as the *multiclass* SVMs [11] and *structured* SVMs [2, 3, 12], respectively. The SVM has serval prominent features. First, it has been proven that maximizing the margin is equivalent to minimising an upper bound of generalization errors [7]. Second, the optimization problem of SVM is convex, which is guaranteed to have a global optimal solution. Third, the size of SVM model is determined by the number of support vectors [7] which is learned from training data, instead of a fixed design in RNNs. However, SVMs are shallow architectures, whereas deep architectures of feedforward and recurrent neural networks have shown state-of-the-art performances in speech recognition [4, 13–15]. Recently deep learning using SVMs in combination with CNNs/DNNs has shown promising results [16, 17]. In this work, a novel model using SVMs in combination with RNNs is proposed.

Conventional RNNs use the softmax active function at the last layer for classification. This work illustrates the advantage of replacing the softmax layer with SVMs. Two training algorithms are proposed, at frame and sequence-level, to jointly learn the parameters of SVM and RNN in the max-margin criteria. The resulting model is termed Recurrent SVM. Conventional SVMs use a predefined feature space and do not have internal states to deal with arbitrary long-term dependencies. The proposed Recurrent SVM uses RNNs to learn the feature space and to capture temporal dependencies, while using the SVM (in the last layer) for classification. Its decoding process is similar to the RNN-HMM hybrid system but with frame-level posterior probabilities replaced by scores from the SVM. We verify its effectiveness on the Windows phone task for speech recognition.

2. RECURRENT NEURAL NETWORK

Given an input sequence $\boldsymbol{\mathcal{X}}_{1:T} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, the simple RNNs compute the hidden vector $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T\}$ by iterating the following equations from $t = 1, \ldots, T$,

$$\boldsymbol{h}_t = \mathcal{H}(\mathbf{W}_{xh}\boldsymbol{x}_t + \mathbf{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \tag{1}$$

where $\mathbf{W}$ is the weight matrix, e.g., $\mathbf{W}_{xh}$ is the input-hidden weight matrix, $\boldsymbol{b}$ is the bias vector, e.g., $\boldsymbol{b}_h$ is the hidden bias vector, and $\mathcal{H}(\cdot)$ is the recurrent hidden layer function. Denote the corresponding state labels as $\boldsymbol{s} = \{s_1, \ldots, s_T\}$, the state-label posterior for frame t is computed by the softmax

$$P(s_t|\boldsymbol{\mathcal{X}}_{1:t}) = \frac{\exp\left(\mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t + b_{s_t}\right)}{\sum_{s_t=1}^{N}\exp\left(\mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t + b_{s_t}\right)} \tag{2}$$

where N is the total number of labels, $\mathbf{w}_{s_t}$ is the weight vector connecting the hidden layer to the output state s_t.[1]

$\mathcal{H}(\cdot)$ in equation (1) is typically an element-wise sigmoid function. However, [18] found that using the Long Short-Term Memory (LSTM) architecture can alleviate the gradient vanishing/exploding issue. A simple LSTM memory block is illustrated in Figure 1. In LSTM-RNN, the recurrent hidden layer function $\mathcal{H}(\cdot)$ is implemented by the following composite functions,

$$
\begin{aligned}
\boldsymbol{z}_t &= \tanh(\mathbf{W}_{zx}\boldsymbol{x}_t + \mathbf{W}_{zh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_z) && \text{block input}\\
\boldsymbol{i}_t &= \sigma\left(\mathbf{W}_{ix}\boldsymbol{x}_t + \mathbf{W}_{ih}\boldsymbol{h}_{t-1} + \mathbf{W}_{ic}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i\right) && \text{input gate}\\
\boldsymbol{f}_t &= \sigma\left(\mathbf{W}_{fx}\boldsymbol{x}_t + \mathbf{W}_{fh}\boldsymbol{h}_{t-1} + \mathbf{W}_{fc}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f\right) && \text{forget gate}\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \boldsymbol{z}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} && \text{cell state}\\
\boldsymbol{o}_t &= \sigma\left(\mathbf{W}_{ox}\boldsymbol{x}_t + \mathbf{W}_{oh}\boldsymbol{h}_{t-1} + \mathbf{W}_{oc}\boldsymbol{c}_{t-1} + \boldsymbol{b}_o\right) && \text{output gate}\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) && \text{block output}
\end{aligned}
\tag{3}
$$

where vectors $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t$ are the activation of the input, output, and forget gates respectively, $\mathbf{W}_{\cdot x}$ are the weight matrices for input $\boldsymbol{x}_t$, $\mathbf{W}_{\cdot h}$ are the weight matrices for recurrent input $\boldsymbol{h}_t$. $\mathbf{W}_{\cdot c}$ are the *diagonal* weight matrices for the peephole connections. $\tanh(\cdot)$, $\sigma(\cdot)$ and $\odot$ are the hyperbolic tangent, logistic sigmoid and multiplication, element-wise functions, respectively.

3. RECURRENT SVM

In LSTM-RNN, the softmax normalization term (Eq. (2)) is a constant for all labels, thus, it can be ignored during decoding. For example, given a input sequence $\boldsymbol{\mathcal{X}}_{1:t}$, the "most likely" state label at time t can be inferred by

$$\arg\max_s \log P(s|\boldsymbol{\mathcal{X}}_{1:t}) = \arg\max_s \mathbf{w}_s^\mathsf{T}\boldsymbol{h}_t \tag{4}$$

For multiclass SVM [11], the classification function is

$$\arg\max_s \mathbf{w}_s^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_t) \tag{5}$$

where $\boldsymbol{\phi}(\boldsymbol{x}_t)$ is the predefined feature space of SVM and $\mathbf{w}_s$ is the weight parameter for class/state s. If RNNs are used to derive the feature space, e.g., $\boldsymbol{\phi}(\boldsymbol{x}_t) \triangleq \boldsymbol{h}_t$, decoding of multiclass SVMs and RNNs are the same. This inspires us to replace the softmax layer in RNNs with SVM. The resulting model is named *Recurrent SVM*. Its architecture is illustrated in Figure 1.

Note that RNNs can be trained using frame-level cross-entropy, connectionist temporal classification (CTC) [19] or sequence-level MMI/sMBR criteria [15,20]. In this work, two training algorithms at frame-level (Section 3.1) and sequence-level (Section 3.2), using max-margin criterion, are proposed. Each algorithm includes two iterative steps. The first step is to estimate the parameters of SVM in the last layer using the quadratic programing. The second step is to update the parameters of RNN in all previous layers using subgradient approaches.

[1]For simplicity, the state bias b_{s_t} is ignored for the rest of the paper.

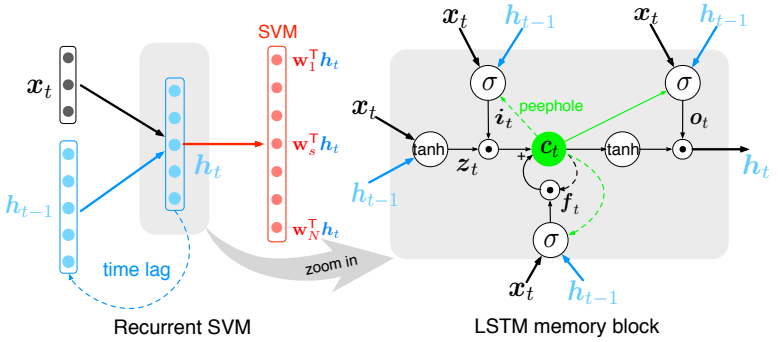

Fig. 1. The architecture of Recurrent SVMs with LSTM units. The dash arrows illustrate the connection with time lag. The SVM in the last layer can be Multiclass/Structured SVMs.

3.1. Frame-level training (Recurrent multiclass SVM)

In the frame-level max-margin training, given training inputs and state labels, $\{(\boldsymbol{x}_t, s_t)\}_{t=1}^T$, let $\boldsymbol{\phi}(\boldsymbol{x}_t) \triangleq \boldsymbol{h}_t$ as the feature space derived from RNN recurrent states, the parameters of the last layer are first estimated using the multiclass SVM training algorithm [11],

$$
\begin{aligned}
\min_{\mathbf{w}_s,\xi_t} \quad & \frac{1}{2}\sum_{s=1}^N \|\mathbf{w}_s\|_2^2 + C\sum_{t=1}^T \xi_t^2 \\
\text{s.t.} \quad & \text{for every training frame } t = 1,\ldots,T, \\
& \text{for every competing states } \bar{s}_t \in \{1,\ldots,N\}: \\
& \mathbf{w}_{s_t}^\mathsf{T}\boldsymbol{h}_t - \mathbf{w}_{\bar{s}_t}^\mathsf{T}\boldsymbol{h}_t \geq 1 - \xi_t, \qquad \bar{s}_t \neq s_t
\end{aligned}
\tag{6}
$$

where $\xi_t \geq 0$ is the slack variable which penalizes the data points that violate the margin requirement. The equation (6) basically says that, the score of the correct state label, $\mathbf{w}_{s_t}^\mathsf{T}\boldsymbol{h}_t$, has to be greater than the scores of any other states, $\mathbf{w}_{\bar{s}_t}^\mathsf{T}\boldsymbol{h}_t$, by a margin. Substituting ξ_t from the constraints into the objective function, equation (6) can be reformulated as minimizing

$$\mathcal{F}_{\texttt{frm}}(\mathbf{w}_s, \mathbf{W}_*) = \frac{1}{2}\sum_{s=1}^N \|\mathbf{w}_s\|_2^2 + C\sum_{t=1}^T \left[1 - \mathbf{w}_{s_t}^\mathsf{T}\boldsymbol{h}_t + \max_{\bar{s}_t \neq s_t} \mathbf{w}_{\bar{s}_t}^\mathsf{T}\boldsymbol{h}_t\right]_+^2 \tag{7}$$

where $\boldsymbol{h}_t$ depends on all the weights $\mathbf{W}_*$ in previous layers shown in equation (3). $[x]_+ = \max(0, x)$ is a hinge function. Note the maximum of a set of linear functions is convex, thus equation (7) is convex w.r.t. $\mathbf{w}_s$. The first step of recurrent SVM training is to optimize $\mathbf{w}_s$ by minimizing (7).

The second step of recurrent SVM training is to optimize the parameters in previous layers, e.g., $\mathbf{W}_{ih}$ in equation (3). These parameters can be updated by back propagating the gradients from the top layer.

$$\frac{\partial \mathcal{F}_{\texttt{frm}}}{\partial \mathbf{W}_{ih}} = \sum_{t=1}^T \left(\frac{\partial \mathcal{F}_{\texttt{frm}}}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \mathbf{W}_{ih}}\right) \tag{8}$$

Note $\partial \boldsymbol{h}_t / \partial \mathbf{W}_{ih}$ is the same as standard RNNs which can be computed using BPTT algorithm. The key here is to compute the derivative of $\mathcal{F}_{\texttt{frm}}$ w.r.t. the $\boldsymbol{h}_t$. However, equation (7) is

not differentiable because of the $\max(\cdot)$. To handle this, the subgradient method [21] is applied. Given the current SVM parameters, $\mathbf{w}_s$, in the last layer, the subgradient of objective function (7) w.r.t. $\boldsymbol{h}_t$ can be derived as

$$\frac{\partial \mathcal{F}_{\texttt{frm}}}{\partial \boldsymbol{h}_t} = 2C\left[1 + \mathbf{w}_{\bar{s}_t}^{\mathsf{T}}\boldsymbol{h}_t - \mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t\right]_+ (\mathbf{w}_{\bar{s}_t} - \mathbf{w}_{s_t}) \quad (9)$$

where $\bar{s}_t = \arg\max_{\bar{s}_t} \mathbf{w}_{\bar{s}_t}^{\mathsf{T}}\boldsymbol{h}_t$ is the most competing state label. After this point, the BPTT algorithm works exactly the same as the standard RNNs. Note, after SVM training for current iteration, most of data may be classified correctly and beyond the margin, i.e., $\left[1 + \mathbf{w}_{\bar{s}_t}^{\mathsf{T}}\boldsymbol{h}_t - \mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t\right]_+ = 0$ for these frames. Interestingly, this means, only the data located in the margin (known as the support vectors) have non-zeros gradients.

3.2. Sequence-level training (Recurrent structured SVM)

In the max-margin sequence training, for simplicity, let's consider one training utterance $(\boldsymbol{\mathcal{X}}, \boldsymbol{s})$ where $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ is the inputs and $\boldsymbol{s} = \{s_1, \ldots, s_T\}$ is reference state labels. The parameters of the model can be estimated by maximising

$$\min_{\bar{\boldsymbol{s}} \neq \boldsymbol{s}} \left\{ \log \frac{P(\boldsymbol{s}|\boldsymbol{\mathcal{X}})}{P(\bar{\boldsymbol{s}}|\boldsymbol{\mathcal{X}})} \right\} = \min_{\bar{\boldsymbol{s}} \neq \boldsymbol{s}} \left\{ \log \frac{p(\boldsymbol{\mathcal{X}}|\boldsymbol{s})P(\boldsymbol{s})}{p(\boldsymbol{\mathcal{X}}|\bar{\boldsymbol{s}})P(\bar{\boldsymbol{s}})} \right\}$$

Here the margin is defined as the minimum distance between the reference state sequence $\boldsymbol{s}$ and competing state sequence $\bar{\boldsymbol{s}}$ in the log posterior domain as illustrated in the Fig. 2. Note that, unlike MMI/sMBR sequence training, the normalization term $\sum_{\boldsymbol{s}} p(\boldsymbol{\mathcal{X}}, \boldsymbol{s})$ in posterior probability is cancelled out, as it appears in both numerator and denominator. For clarity, the language model probability is not shown here. To generalize the above objective function, a loss function $\mathcal{L}(\boldsymbol{s}, \bar{\boldsymbol{s}})$ is introduced to control the size of the margin, a hinge function $[\cdot]_+$ is applied to ignore the data that beyond the margin, and a prior $P(\mathbf{w})$ is incorporated to further reduce the generalization error. Thus the criterion becomes minimizing

$$-\log P(\mathbf{w}) + \left[\max_{\bar{\boldsymbol{s}} \neq \boldsymbol{s}} \left\{ \mathcal{L}(\boldsymbol{s}, \bar{\boldsymbol{s}}) - \log \frac{p(\boldsymbol{\mathcal{X}}|\boldsymbol{s})P(\boldsymbol{s})}{p(\boldsymbol{\mathcal{X}}|\bar{\boldsymbol{s}})P(\bar{\boldsymbol{s}})} \right\}\right]_+^2 \quad (10)$$

For recurrent SVMs, $\log\left(p(\boldsymbol{\mathcal{X}}|\boldsymbol{s})P(\boldsymbol{s})\right)$ can be computed via

$$\sum_{t=1}^{T} \left(\mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t - \log P(s_t) + \log P(s_t|s_{t-1})\right) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{\mathcal{X}}, \boldsymbol{s}) \quad (11)$$

where $\boldsymbol{\phi}(\boldsymbol{\mathcal{X}}, \boldsymbol{s})$ is the joint feature [22], which characterizing the dependencies between two sequences, $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{s}$,

$$\boldsymbol{\phi}(\boldsymbol{\mathcal{X}}, \boldsymbol{s}) = \sum_{t=1}^{T} \begin{bmatrix} \delta(s_t = 1)\boldsymbol{h}_t \\ \vdots \\ \delta(s_t = N)\boldsymbol{h}_t \\ \log P(s_t) \\ \log P(s_t|s_{t-1}) \end{bmatrix}, \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_N \\ -1 \\ +1 \end{bmatrix} \quad (12)$$

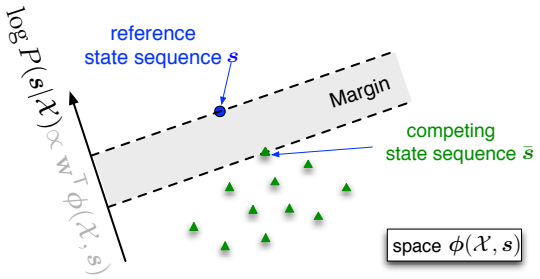

Fig. 2. The sequence-level max-margin criterion. Margin is defined in the log-postieror domain between the reference state sequence S and the most competing state sequence $\bar{\boldsymbol{s}}$.

where $\delta(\cdot)$ is the Kronecker delta (indicator) function. Here the prior, $P(\mathbf{w})$, is assumed to be a Gaussian with a zero mean and a scaled identity covariance matrix $C\mathbf{I}$, thus $\log P(\mathbf{w}) = \log \mathcal{N}(\mathbf{0}, C\mathbf{I}) \propto -\frac{1}{2C}\mathbf{w}^{\mathsf{T}}\mathbf{w}$. Substituting the prior and equation (11) into criterion (10), the parameters of recurrent SVM can be estimated by minimizing[2]

$$\mathcal{F}_{\texttt{seq}}(\mathbf{w}, \mathbf{W}_*) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\Bigg[\overbrace{-\mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{\mathcal{X}}, \boldsymbol{s})}^{\text{linear}} + \underbrace{\max_{\bar{\boldsymbol{s}} \neq \boldsymbol{s}} \left\{ \mathcal{L}(\boldsymbol{s}, \bar{\boldsymbol{s}}) + \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{s}}) \right\}}_{\text{convex}}\Bigg]_+^2 \quad (13)$$

where $\boldsymbol{\phi}(\cdot, \cdot)$ depends on $\boldsymbol{h}_t$ and $\boldsymbol{h}_t$ depends on all the weights $\mathbf{W}_*$ in previous layers shown in equation (3). The first step of sequence training is to optimize $\mathbf{w} = \{\mathbf{w}_1, \ldots, \mathbf{w}_s, \ldots, \mathbf{w}_N\}$ in the last layer by minimizing (13). Similar to $\mathcal{F}_{\texttt{frm}}$, the $\mathcal{F}_{\texttt{seq}}$ is also convex for $\mathbf{w}$. Interestingly, given the feature space $\boldsymbol{\phi}$, equation (13) is the same as the training criterion for structured SVMs [12]. This optimization can be solved using the cutting plane algorithm [2].

The second step of sequence training is to optimize the parameters $\mathbf{W}_*$ in the previous layers, e.g., $\mathbf{W}_{ih}$ in equation (3). These parameters can be updated by back propagating the gradients from the top layer.

$$\frac{\partial \mathcal{F}_{\texttt{seq}}}{\partial \mathbf{W}_{ih}} = \sum_{t=1}^{T} \left(\frac{\partial \mathcal{F}_{\texttt{seq}}}{\partial \boldsymbol{h}_t} \frac{\partial \boldsymbol{h}_t}{\partial \mathbf{W}_{ih}} \right) \quad (14)$$

Again $\partial \boldsymbol{h}_t / \partial \mathbf{W}_{ih}$ is the same as standard RNNs. The key is to calculate the subgradient of $\mathcal{F}_{\texttt{seq}}$ w.r.t. $\boldsymbol{h}_t$ for each frame t,

$$\frac{\partial \mathcal{F}_{\texttt{seq}}}{\partial \boldsymbol{h}_t} = 2C\left[\mathcal{L} + \mathbf{w}^{\mathsf{T}}\bar{\boldsymbol{\phi}} - \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}\right]_+ (\mathbf{w}_{\bar{s}_t} - \mathbf{w}_{s_t}) \quad (15)$$

where $\mathcal{L}$ is the loss between the reference $\boldsymbol{s}$ and its most competing state sequence $\bar{\boldsymbol{s}}$, and $\bar{\boldsymbol{\phi}}$ is short for feature $\boldsymbol{\phi}(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{s}})$. After this point, the BPTT algorithm works exactly the same as the standard RNNs. The term $\left[\mathcal{L} + \mathbf{w}^{\mathsf{T}}\bar{\boldsymbol{\phi}} - \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}\right]_+$ in equation (15) indicates that, only the utterances located in the margin (support vectors) have non-zeros gradients.

[2]For simplicity, only one training utterance is shown in equation (13).

3.3. Inference

The decoding process is similar to the RNN-HMM hybrid system but with the log posterior probabilities, $\log P(s_t|\boldsymbol{x}_t)$, replaced by the scores from recurrent SVM, $\mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t$. Note that search the most likely state sequence $\boldsymbol{s}$ during decoding is essentially the same as search the most competing state sequence $\bar{\boldsymbol{s}}$ in equation (13), except the loss $\mathcal{L}(\boldsymbol{s}, \bar{\boldsymbol{s}})$. They can be solved using the same Viterbi algorithm.

3.4. Practical Issues

An efficient implementation of the algorithm is important for speech recognition. In this section several design options for recurrent SVM training are described that have a substantial influence on computational efficiency.

Form of Prior. Previously a zero-mean Gaussian prior is introduced. However, a proper mean of prior should be the one can yield LSTM performance, $\log P(\mathbf{w}) = \log \mathcal{N}(\mathbf{w}_{\texttt{RNN}}, C\mathbf{I})$. Thus the term $\frac{1}{2}\|\mathbf{w}\|_2^2$ in (7) and (13) becomes $\frac{1}{2}\|\mathbf{w}-\mathbf{w}_{\texttt{RNN}}\|_2^2$. Since a better mean is applied, a smaller variation C can be used to reduce the training iterations [23].

Lattices. Solving the optimization (13) requires to search the most competing state sequence $\bar{\boldsymbol{s}}$ efficiently. The computational load during training is dominated by this search process. To speed up the training, denominator lattices with state alignments are used to constraint the searching space. Then a lattice-based forward-backward search [3, 23] is applied to find the most competing state sequence $\bar{\boldsymbol{s}}$.

Caching. To further avoid the computation cost of searching the $\bar{\boldsymbol{s}}$ in (13), during training iterations, the five most recently used state sequences $\bar{\boldsymbol{s}}$ for each utterance are cached. This reduces the number of calls to search in the lattices.

4. EXPERIMENTS

The Windows Phone short message diction task is used to evaluate the effectiveness of the recurrent SVM proposed in Section 3. The training data consists of 60 hours of transcribed US-English speech. The test set consists of 3 hours of data and 16k words. All the experiments were implemented based on the computational network toolkit (CNTK) [24]. 87 dimentional log-filter-bank features are used for LSTM and recurrent SVM. The feature context window is 11 frames. The baseline DNN has 5 hidden layers, each layer includes 2048 hidden nodes. All the systems use 1812 tied triphine states. The baseline LSTM has four layers, each has 1024 hidden nodes and the output size of each layer is reduced to 512 using a linear projection layer [25]. The state label is delayed by 5 frames for LSTM as described in [25]. No frame stacking was applied. During LSTM training, the back propagation though time (BPTT) step is set to 20. The forget gate bias $\boldsymbol{b}_f$ in (3) is initialized as $\mathbf{1}$ according to [26].

The Recurrent SVM is constructed base on the baseline LSTM (four LSTM layers) in combination with a SVM in the last softmax layer. In the frame-level training, the scaled 0/1 loss is used in equation (6) and the CE trained LSTM is used as a prior. The scalar is tuned using the cross validation set. In the sequence training, the state loss [15] is applied in equation (13) and the MMI-trained LSTM is used as a prior. The results of all systems are shown in Table 1. In frame-level training, the proposed recurrent SVM improved the LSTM (CE) by 2.1%, and improved the DNN (CE) by 10.3%. In sequence-level training, the recurrent SVM outperformed the LSTM (MMI) by 2.8%, and improved the DNN (MMI) by 6%. Table 2 suggests that in both training criterion, more than 65% of gains in recurrent SVMs are coming from updating the recurrent layers. Note that although the improvement of recurrent SVM over LSTM is not big, it does not increase any latency/computation cost for the runtime decoding.

Model	**WER %**	
	frame-level training	sequence training
DNN	23.06% (CE)	21.08 % (MMI)
LSTM-RNN	21.14% (CE)	20.40% (MMI)
Recurrent SVM	20.69% (MM)	19.83% (MM)

Table 1. The results (in word error rate) for DNN, LSTM-RNN and Recurrent SVM systems, trained using frame-level and sequence-level criterion. The CE, MMI and MM are short for cross entropy (CE), maximum mutual information (MMI) and maximum margin (MM) criterion.

Training	**Recurrent SVM**		**LSTM**
	softmax layer	+ previous layers	baseline
frame-level	21.01 %	20.69%	21.14%
sequence-level	20.19%	19.83%	20.40%

Table 2. The impact of each layer in recurrent SVM using the frame-level and sequence-level max-margin training.

5. CONCULUTION AND FUTURE WORK

A new type of recurrent model is described in this work. The traditional RNN uses the softmax activation at the last layer for classification. The proposed model instead uses a SVM at the last layer. Two training algorithms are proposed at frame and sequence-level to learn parameters of the SVM and RNN jointly in maximum-margin criteria. In frame-level training, the new model is shown to be related to the multiclass SVM with RNN features; In sequence-level training, it is related to the structured SVM with RNN features and state transition features. The proposed model, named recurrent SVMs, yields 2.8% improvement over LSTM (MMI trained) on Windows Phone task without increasing any runtime latency. Unlike the conventional SVMs need predefined feature space, the proposed recurrent SVM uses LSTMs to learn the feature space and to capture temporal dependencies, while using the linear SVM in the last layer for classification. Future work will investigate recurrent SVMs with non-linear kernels.

6. REFERENCES

[1] C. M. Bishop, *Pattern recognition and machine learning*, Springer, 2006.

[2] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural SVMs," *Journal of Machine Learning Research*, vol. 77, pp. 27–59, 2009.

[3] S.-X. Zhang and M. J. F. Gales, "Structured SVMs for automatic speech recognition," *IEEE Transactions Audio, Speech and Language Processing*, vol. 21, no. 3, pp. 544–555, 2013.

[4] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*. IEEE, 2013, pp. 6645–6649.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[6] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst*, vol. 2, pp. 303–314, 1989.

[7] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

[8] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[9] T. Joachims, "Text categorization with suport vector machines: Learning with many relevant features," in *Proceedings of 10th European Conference on Machine Learning*, 1998, pp. 137–142.

[10] J. Salomon, S. King, and M. Osborne, "Framewise phone classification using support vector machines," in *Proceedings of Interspeech*, 2002, pp. 2645–2648.

[11] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," vol. 2, pp. 265–292, 2001.

[12] S.-X. Zhang and M. J. F. Gales, "Structured support vector machines for noise robust continuous speech recognition," in *Proceedings of Interspeech*, Florence, Italy, 2011, pp. 989–992.

[13] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[14] G. Hinton, L. Deng, D. Yu, G. E. Dahl, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[15] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *INTERSPEECH*, 2013, pp. 2345–2349.

[16] Y. Tang, "Deep learning using linear support vector machines," in *Internaltion Conference on Machine Learning*, December 2013.

[17] S.-X. Zhang, C. Liu, K. Yao, and Y. Gong, "Deep neural support vector machines for speech recognition," in *ICASSP*. IEEE, April 2015, pp. 4275–4279.

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

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006, pp. 369–376.

[20] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *ICASSP*. IEEE, 2015, pp. 4280–4284.

[21] Y. Singer and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for SVM," in *Proceedings of ICML*, 2007, pp. 807–814.

[22] S.-X. Zhang, A. Ragni, and M. J. F. Gales, "Structured log linear models for noise robust speech recognition," *Signal Processing Letters, IEEE*, vol. 17, pp. 945–948, 2010.

[23] S.-X. Zhang, *Structured Support Vector Machines for Speech Recogntion*, Ph.D. thesis, Cambridge University, March 2014.

[24] D. Yu, A. Eversole, M. Seltzer, K. Yao, et al., "An introduction to computational networks and the computational network toolkit," Tech. Rep. MSR-TR-2014-112, Microsoft Technical Report, 2014.

[25] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH*, 2014.

[26] R. Jozefowicz, W. Zaremba, and L. Sutskever, "An empirical exploration of recurrent network architectures," in *ICML*, 2015, pp. 2342–2350.