DEEP NEURAL SUPPORT VECTOR MACHINES FOR SPEECH RECOGNITION

Shi-Xiong Zhang, Chaojun Liu, Kaisheng Yao and Yifan Gong

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

{zhashi, chaojunl, kaisheny, ygong}@microsoft.com

ABSTRACT

A new type of deep neural networks (DNNs) is presented in this paper. Traditional DNNs use the multinomial logistic regression (softmax activation) at the top layer for classification. The new DNN instead uses a support vector machine (SVM) at the top layer. Two training algorithms are proposed at the frame and sequence-level to learn parameters of SVM and DNN in the maximum-margin criteria. In the frame-level training, the new model is shown to be related to the multiclass SVM with DNN features; In the sequence-level training, it is related to the structured SVM with DNN features and HMM state transition features. Its decoding process is similar to the DNN-HMM hybrid system but with frame-level posterior probabilities replaced by scores from the SVM. We term the new model deep neural support vector machine (DNSVM). We have verified its effectiveness on the TIMIT task for continuous speech recognition.

Index Terms— DNN, multiclass SVM, structured SVM, maximum margin, sequence training

1. INTRODUCTION

Neural Networks and Support Vector Machines (SVMs) are two successful approaches for supervised machine learning and classification [1]. Neural Networks are universal models in the sense that they can approximate any nonlinear functions arbitrary well on a compact interval [2]. However, there are two major drawbacks of neural networks. 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].

Alternatively, the SVMs [3] supplied with the maximum margin classifier idea has received extensive research attentions [4–6]. The SVM has serval prominent features. First, it has been proven that maximizing the margin is equivalent to minimising an upper bound on the generalization error [3]. Second, the optimization problem of SVM is convex, which is guaranteed to have a global optimal solution. The SVM was originally proposed for binary classification. It can be extended to handle the multiclass classification or sequence recognition using the majority voting [7] or directly modifying the optimization [8–10].[1] However, SVMs are in principle shallow architectures, whereas deep architectures with neural networks have been shown to achieve state-of-the-art performances in speech recognition [13–15]. Although there are some works on deep architectures of binary SVMs [16, 17], how to extend them for ASR is still open problem.

This paper is a first attempt (to the best of our knowledge) on deep learning using SVM for ASR. Traditional deep neural networks use the multinomial logistic regression (softmax activation function) at the top layer for classification. This work illustrates the advantage of replacing the logistic regression with a SVM. Two training algorithms are proposed at frame and sequence-level to learn the parameters of SVM and DNN in maximum-margin criteria. In the frame-level training, the new model is shown to be related to the multiclass SVM with DNN features; In the sequence-level training, it is related to the structured SVM with DNN features and HMM state transition features [18, 19]. In the sequence case, the parameters of SVM, HMM state transitions and language models can be jointly learned. Its decoding process is similar to the DNN-HMM hybrid system but with frame-level posterior probabilities replaced by scores from the SVM. We term the new model deep neural support vector machine (DNSVM) and verify its effectiveness on the TIMIT task for continuous speech recognition.

2. DEEP NEURAL SVM

Most of the DNNs use the multinomial logistic regression, also known as softmax active function, at the top layer for classification. Specifically, given the observation $\boldsymbol{o}_t$ at frame t, let $\boldsymbol{h}_t$ is the output vector of the top hidden layer in DNNs, the output of DNNs for state s_t can be expressed as

$$P(s_t|\boldsymbol{o}_t) = \frac{\exp\left(\mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t\right)}{\sum_{s_t=1}^{N}\exp\left(\mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t\right)} \quad (1)$$

where $\mathbf{w}_{s_t}$ are the weights connecting the last hidden layer to the output state s_t, and N is the number of states. Note the normalization term in equation (1) is independent of states,

[1]The modified SVMs for multiclass classification and sequence recognition are known as the multiclass SVMs [8] and structured SVMs [9, 11, 12], respectively.

thus, it can be ignored during frame classification or sequence decoding.[2] For example, in the frame classification, given an observation $\boldsymbol{o}_t$, the corresponding state s_t can be inferred by

$$\arg\max_s \log P(s|\boldsymbol{o}_t) = \arg\max_s \mathbf{w}_s^\top \boldsymbol{h}_t \tag{2}$$

For multiclass SVM [8], the classification function is

$$\arg\max_s \mathbf{w}_s^\top \boldsymbol{\phi}(\boldsymbol{o}_t) \tag{3}$$

where $\boldsymbol{\phi}(\boldsymbol{o}_t)$ is the predefined feature space and $\mathbf{w}_s$ is the weight parameter for class/state s. If DNNs are used to derive the feature space, e.g., $\boldsymbol{\phi}(\boldsymbol{o}_t) \triangleq \boldsymbol{h}_t$, decoding of multiclass SVMs and DNNs are the same. Note that DNNs can be trained using the frame-level cross-entropy (CE) or sequence-level MMI/sMBR criteria [15]. In this paper, two algorithms, at frame and sequence-level, are also proposed to estimate the parameters of SVM (in the last layer) and to update the parameters of DNN (in all previous layers) using maximum margin criteria. The resulting model is named Deep Neural SVM (DNSVM). Its architecture is illustrated in Fig. 1.

2.1. Frame-level max-magin training

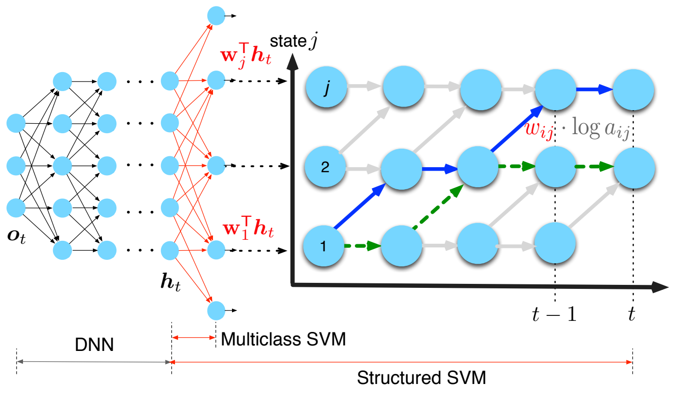
Given the training observations and their corresponding state labels, $\{(\boldsymbol{o}_t, s_t)\}_{t=1}^T$, where $s_t \in \{1,\dots,N\}$, in frame-level training, the parameters of DNNs are normally estimated by minimizing the cross-entropy. In this work, let $\boldsymbol{\phi}(\boldsymbol{o}_t) \triangleq \boldsymbol{h}_t$ as the feature space derived from the DNN, the parameters of the last layer are first estimated using the multiclass SVM training algorithm [8],

$$\begin{aligned}\min_{\mathbf{w}_s,\xi_t}\quad & \frac{1}{2}\sum_{s=1}^N \|\mathbf{w}_s\|_2^2 + C\sum_{t=1}^T \xi_t^2 \qquad (4)\\ \text{s.t.}\quad & \text{for every training frame } t=1,\dots,T,\\ & \text{for every competing states } \bar{s}_t \in \{1,\dots,N\}:\\ & \mathbf{w}_{s_t}^\top \boldsymbol{h}_t - \mathbf{w}_{\bar{s}_t}^\top \boldsymbol{h}_t \geq 1-\xi_t, \qquad \bar{s}_t \neq s_t\end{aligned}$$

where $\xi_t \geq 0$ is the slack variable which penalizes the data points that violate the margin requirement. Note that the objective function is essentially the same as the binary SVM. The only difference comes from the constraints, which basically says that, the score of the correct state label, $\mathbf{w}_{s_t}^\top \boldsymbol{h}_t$, has to be greater than the scores of any other states, $\mathbf{w}_{\bar{s}_t}^\top \boldsymbol{h}_t$, by a margin determined by the loss. In equation (4) the loss is a constant 1 for any misclassification. According to [16], using the squared slacks is slightly better than ξ_t, thus ξ_t^2 is applied in equation (4).

Note if the correct score, $\mathbf{w}_{s_t}^\top \boldsymbol{h}_t$, is greater than all the competing scores, $\mathbf{w}_{\bar{s}_t}^\top \boldsymbol{h}_t$, it must be greater than the "most" competing score, $\max_{\bar{s}_t \neq s_t} \mathbf{w}_{\bar{s}_t}^\top \boldsymbol{h}_t$. Thus, substituting the slack

[2]Thoeritically, the softmax normalization can be ignored without affecting the performance, but it may affect the pruning.



$\mathbf{w}_j^\top \boldsymbol{h}_t$
state j
j
2
1
$w_{ij} \cdot \log a_{ij}$
$\boldsymbol{o}_t$
$\mathbf{w}_1^\top \boldsymbol{h}_t$
$\boldsymbol{h}_t$
$t-1$
t
DNN
Multiclass SVM
Structured SVM

Fig. 1. The architecture of Deep Neural SVMs. The double-headed arrows illustrate the scope of parameters for DNNs, Multiclass SVMs and Structured SVMs. For sequence-level max-margin training, the solid blue arrows (in the trellis) represent the reference state sequence, and the dash green arrows represent the most competing state sequence.

variable ξ_t from the constraints into the objective function, equation (4) can be reformulated as the minimization of

$$\mathcal{F}_{\texttt{fMM}}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{t=1}^T \left[1 - \mathbf{w}_{s_t}^\top \boldsymbol{h}_t + \max_{\bar{s}_t \neq s_t} \mathbf{w}_{\bar{s}_t}^\top \boldsymbol{h}_t\right]_+^2 \tag{5}$$

where $\mathbf{w} = [\mathbf{w}_1^\top,\dots,\mathbf{w}_N^\top]^\top$ are the parameter vectors for each state and $[\cdot]_+$ is the hinge function [20]. Note the maximum of a set of linear functions is convex, thus equation (5) is convex with respect to $\mathbf{w}$.

Given the multiclass SVM parameters $\mathbf{w}$, the parameters of the previous layer $\mathbf{w}^{[l]}$, can be updated by back propagating the gradients from the top layer multiclass SVM,

$$\frac{\partial \mathcal{F}_{\texttt{fMM}}}{\partial w_i^{[l]}} = \sum_{t=1}^T \left(\frac{\partial \mathcal{F}_{\texttt{fMM}}}{\partial \boldsymbol{h}_t}^\top \frac{\partial \boldsymbol{h}_t}{\partial w_i^{[l]}} \right). \tag{6}$$

Note $\partial \boldsymbol{h}_t / \partial w_i^{[l]}$ is the same as standard DNNs. The key is to compute the derivative of $\mathcal{F}_{\texttt{fMM}}$ w.r.t. the activations, $\boldsymbol{h}_t$. However, equation (5) is not differentiable because of the hinge function and $\max(\cdot)$. To handle this, the subgradient method [21] is applied. Given the current multiclass SVM parameters (in the last layer) for each state, $\mathbf{w}_s$, and the most competing state label $\bar{s}_t = \arg\max_{\bar{s}_t} \mathbf{w}_{\bar{s}_t}^\top \boldsymbol{h}_t$, the subgradient of objective function (5) can be expressed as

$$\frac{\partial \mathcal{F}_{\texttt{fMM}}}{\partial \boldsymbol{h}_t} = 2C\left[1 + \mathbf{w}_{\bar{s}_t}^\top \boldsymbol{h}_t - \mathbf{w}_{s_t}^\top \boldsymbol{h}_t\right]_+ (\mathbf{w}_{\bar{s}_t} - \mathbf{w}_{s_t}) \tag{7}$$

After this point, the backpropagation algorithm is exactly the same as the standard DNNs. Note that, after training of multiclass SVMs, most of training frames can be classified correctly and beyond the margin. This means, for those frames, $\mathbf{w}_{s_t}^\top \boldsymbol{h}_t > \mathbf{w}_{\bar{s}_t}^\top \boldsymbol{h}_t + 1$. Thus, only the rest few training samples (support vectors) have non-zeros sugradients.

2.2. Sequence-level max-margin training

In the max-margin sequence training, for simplicity, first consider one training utterance $(\mathbf{O}, S)$, where $\mathbf{O} = \{\boldsymbol{o}_1, \dots, \boldsymbol{o}_T\}$ is the observation sequence and $S = \{s_1, \dots, s_T\}$ is the corresponding reference states. The parameters of the model can be estimated by maximising

$$\min_{\bar{S} \neq S} \left\{ \log \frac{P(S|\mathbf{O})}{P(\bar{S}|\mathbf{O})} \right\} = \min_{\bar{S} \neq S} \left\{ \log \frac{p(\mathbf{O}|S)P(S)}{p(\mathbf{O}|\bar{S})P(\bar{S})} \right\}$$

Here the margin is defined as the minimum distance between the reference state sequence S and competing state sequence $\bar{S}$ in the log posterior domain as illustrated in the Fig. 2. Note that, unlike MMI/sMBR sequence training, the normalization term $\sum_S p(\mathbf{O}, S)$ 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 $\mathcal{L}(S, \bar{S})$ 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

$$-\log P(\mathbf{w}) + \left[\max_{\bar{S} \neq S} \left\{ \mathcal{L}(S, \bar{S}) - \log \frac{p(\mathbf{O}|S)P(S)}{p(\mathbf{O}|\bar{S})P(\bar{S})} \right\} \right]_+^2 \quad (8)$$

For DNSVM, the $\log\left(p(\mathbf{O}|S)P(S)\right)$ can be computed via

$$\sum_{t=1}^{T} \left(\mathbf{w}_{s_t}^{\mathsf{T}} \boldsymbol{h}_t - \log P(s_t) + \log P(s_t|s_{t-1}) \right) = \mathbf{w}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, S) \quad (9)$$

where $\boldsymbol{\phi}(\mathbf{O}, S)$ is the joint feature [22], which characterizes the dependencies between $\mathbf{O}$ and S,

$$\boldsymbol{\phi}(\mathbf{O}, S) = \sum_{t=1}^{T} \begin{bmatrix} \delta(s_t = 1)\boldsymbol{h}_t \\ \vdots \\ \delta(s_t = N)\boldsymbol{h}_t \\ \log P(s_t) \\ \log P(s_t|s_{t-1}) \end{bmatrix}, \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_N \\ -1 \\ +1 \end{bmatrix} \quad (10)$$

where $\delta(\cdot)$ is the the Kronecker delta (indicator) function.[3] 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 (9) into criterion (8), the parameters of DNSVM (in the last layer) can be estimated by minimizing

$$\mathcal{F}_{\texttt{sMM}}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{u=1}^{U} \Bigg[\overbrace{-\mathbf{w}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}_u, S_u)}^{\text{linear}} \quad (11)$$
$$+ \underbrace{\max_{\bar{S}_u \neq S_u} \left\{ \mathcal{L}(S_u, \bar{S}_u) + \mathbf{w}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O_u}, \bar{S}_u) \right\}}_{\text{convex}} \Bigg]_+^2$$

[3] For clarity, constant weights "-1" and "+1" are shown in (10). In practice the weirghts for state priors and transition probabilities are also learned.

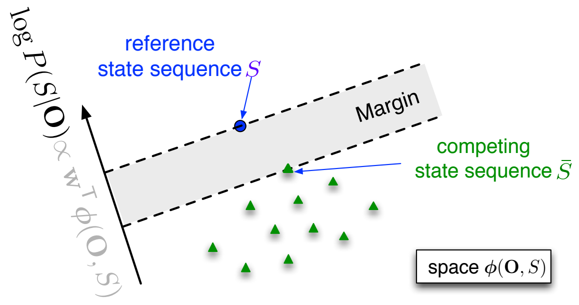

reference
state sequence S
Margin
competing
state sequence $\bar{S}$
space $\phi(\mathbf{O}, S)$
$\log P(S|\mathbf{O}) \propto \mathbf{w}^{\mathsf{T}}\phi(\mathbf{O}, S)$

Fig. 2. The margin is defined in the log-postieria domain between reference state sequence S and the most competing state sequence $\bar{S}$.

where $u = 1, \dots, U$ is the index of training utterances. Like the $\mathcal{F}_{\texttt{fMM}}$, $\mathcal{F}_{\texttt{sMM}}$ is also convex for $\mathbf{w}$. Interestingly, objective function (11) for DNSVM is the same as the training criterion for structured SVMs [9] with the features defined in (10).

To solve equation (11), the cutting plane algorithm [11] can be applied. It requires to search the most competing state sequence $\bar{S}_u$ efficiently. If the state-level loss is applied, the search problem, $\max_{\bar{S}_u}\{\cdot\}$, can be solved using the Viterbi decoding algorithm (see section 2.3). 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 search space. Then a lattice-based forward-backward search [12, 19] is applied to find the most competing state sequence $\bar{S}_u$.

Similar to the frame-level case, the parameters of previous layers can also be updated by back propagating the gradients from the top layer. The key is to calculate the subgradient of $\mathcal{F}_{\texttt{sMM}}$ w.r.t. $\boldsymbol{h}_t$ for utterance u and frame t,

$$\frac{\partial \mathcal{F}_{\texttt{sMM}}}{\partial \boldsymbol{h}_t} = 2C \left[\mathcal{L} + \mathbf{w}^{\mathsf{T}} \bar{\boldsymbol{\phi}} - \mathbf{w}^{\mathsf{T}} \boldsymbol{\phi} \right]_+ (\mathbf{w}_{\bar{s}_t} - \mathbf{w}_{s_t}) \quad (12)$$

where $\mathcal{L}$ is the loss for between the reference S_u and its most competing state sequence $\bar{S}_u$, and $\bar{\boldsymbol{\phi}}$ is short for $\boldsymbol{\phi}(\mathbf{O}_u, \bar{S}_u)$. After this point, the backpropagation algorithm is exactly the same as the standard DNNs. According to [23], fine tuning the previous layers in MMI training does not have gain. The equation (12) is not implemented in this work. It was discussed for theoretical interests.

2.3. Inference

The decoding process is similar to the standard DNN-HMM hybrid system but with posterior probabilities, $\log P(s_t|\boldsymbol{o}_t)$, replaced by the scores from DNSVM, $\mathbf{w}_{s_t}^{\mathsf{T}}\boldsymbol{h}_t$. If the sequence training is applied, the state priors, state transition probabilities (in log domain) and language model scores are also scaled by the weights that learned from equation (11). Note that decoding the most likely state sequence S is essentially the same as inferring the most competing state sequence $\bar{S}_u$ in equation (11), except for the loss $\mathcal{L}(S_u, \bar{S}_u)$. They can be solved using the same Viterbi algorithm (see Fig. 1).

2.4. Practical Issues

An efficient implementation of the algorithm is important for speech recognition. In this section several design options for DNSVM training (at frame and sequence-level) are described that have a substantial influence on computational efficiency.

Form of Prior. Previously a zero-mean Gaussian prior is used in (11). However, a proper mean of prior for DNSVM should be the parameters of DNN, $\log P(\mathbf{w}) = \log \mathcal{N}(\mathbf{w}_{\mathtt{DNN}}, C\mathbf{I})$. Thus the term $\frac{1}{2}\|\mathbf{w}\|_2^2$ in (5) and (11) becomes $\frac{1}{2}\|\mathbf{w}-\mathbf{w}_{\mathtt{DNN}}\|_2^2$. Since a better mean is applied, a smaller variation C can be used to reduce the training iterations [19].

Caching. To avoid the computation cost by repeatedly searching the same $\bar{s}_t$ and $\overline{S}_u$ in (5) and (11) during training iterations, the 5 most recently used $\bar{s}_t$ and $\overline{S}_u$ for each training sample are cached. This reduces the number of calls to search in the full decoding space or lattices.

Parallelization. The computational load during training is dominated by finding the best competing state sequence $\overline{S}_u$ for each utterance. One could make use of up to U parallel threads, each searching the $\overline{S}_u$ for a subset of training data. A central server can be used to collect $\overline{S}_u$ from each thread and then update the parameters.

3. EXPERIMENTS

The TIMIT phone recognition task [24] was used to evaluate the effectiveness of the deep neural SVM proposed in Section 2. The training set of the TIMIT corpus contains 462 speakers and 3696 utterances. A separate dev set (400 utterances) was used for tuning the hyper parameters, e.g. the penalty factor C in equations (5) and (11). Results are reported using the core test set (192 utterances). Fourier transform based log filterbank with 40 coefficients and log energy (distributed on a Mel scale) along with deltas and double deltas were extracted to form 123-dimensional observations.

To be consistent with previous works [25], 183 target class labels were used with 3 states for each of 61 phones. After decoding, the original 61 context-independent phones were mapped to a set of 39 phones for final scoring according to the standard evaluation protocol. A bi-gram language model over phones, estimated from the training set, was used in decoding and sequence training. To prepare the DNN training, a triphone based GMM-HMM system was built to produce state-level labels using Viterbi alignment. The DNN had three layers of 2000 hidden units each, 15-frame context input, and was trained using the CE criterion. The baseline results of GMM and DNN-HMM are shown in Table 1. They are comparable with results produced by state-of-the-art open-source tools, e.g., CNTK [26].

GMM		DNN	DNSVM	
monophone	triphone	CE	frame MM	seq. MM
31.02%	26.23%	22.87%	21.95%	21.04%

Table 1. The results (in phone error rate) for GMM-HMM, DNN-HMM and DNSVM-HMM systems trained using maximum likelihood, cross-entropy (CE) and max-margin (MM) criteria, respectively. All the systems use the same labels representing 61 monophones with 3 states per phone.

DNN Features $\boldsymbol{h}_t$	DNSVM (frame-level)	
	last layer only	+ previous layers
183 posterior features	22.03%	21.95%
2000 top hidden features	21.90%	21.90%

Table 2. The results (in phone error rate) of DNSVMs using frame-level max-margin training.

DNN Features $\boldsymbol{h}_t$	DNSVM (sequence-level)	
	acoustic only	joint learn with LM
183 posterior features	21.38%	21.04%

Table 3. The performance of DNSVM using sequence-level max-margin training without updating previous layers.

For DNSVM, in the frame-level training, the scaled 0/1 loss was used in equation (4) instead of "1". The scalar was tuned using the dev set. In the sequence training, the state loss [15] was applied in equation (11). Although top hidden layer features $\boldsymbol{h}_t$ were used to derive the DNSVM in section 2, any other forms of DNN features, e.g., posterior features, could also be applied in equations (4) and (11). These features are examined in Table 2. Using top hidden layer features was slightly better, however it required much more memory during training. In the frame-level case, DNSVMs improve over DNNs by 4% relatively. In the sequence case, the weights for each state prior, transition probability and bigram probability can be jointly learned with acoustic parameters (see section 2.2). The results are shown in Table 3. Comparing to the DNN baseline in Table 1, the proposed DNSVM provided 8% relative error rate reduction.

4. CONCULUTION AND FUTURE WORK

We have presented a new type of DNN. Traditional DNNs use the softmax at the top layer for classification. The new DNN instead uses a SVM at the top layer. We have derived training algorithms at the frame and sequence-level to jointly learn parameters of SVM and DNN in the maximum-margin criteria. In the frame-level training, the new model is shown to be related to the multiclass SVM with DNN features; In the sequence training, it is related to the structured SVM with DNN features and state transition features. The proposed model, named DNSVM, yields 8% relative error rate reduction over DNNs (CE trained) on TIMIT task. Future works will examine the DNSVM on top of sequence trained DNNs [15] on large vocabulary tasks. In this work only the last layer of DNNs is replaced by linear SVMs, future work will also investigate non-linear kernels and "DNN-free" deep SVMs.

5. REFERENCES

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

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

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

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

[5] 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.

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

[7] C. Gautier, "Filter trees for combining binary classifiers," M.S. thesis, Cambridge University, 2008.

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

[9] 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.

[10] G. Heigold, P. Dreuw, S. Hahn, R. Schluter, and H. Ney, "Margin-based discriminative training for string recognition," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 6, pp. 917–925, 2010.

[11] 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.

[12] 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.

[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] M.A. Wiering, M. Schutten, A. Millea, A. Meijster, and L. Schomaker, "Deep support vector machines for regression problems," in *International Workshop on Advances in Regularization, Optimization, Kernel Methods, and Support Vector Machines*, 2013, pp. 53–54.

[18] S. V. Ravuri, "Hybrid mlp/structured-svm tandem systems for large vocabulary and robust asr," in *Proceedings of Interspeech*, Sigapore, 2014, pp. 2729–2733.

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

[20] C. Gentile and M. K. Warmuth, "Linear hinge loss and average margin," in *NIPS*, Michael J. Kearns, Sara A. Solla, and David A. Cohn, Eds. 1998, pp. 225–231, The MIT Press.

[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] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior, and M. Bacchiani, "Asynchronous stochastic optimization for sequence training of deep neural networks," in *ICASSP*. IEEE, 2014, pp. 5587–5591.

[24] L. Lamel, R. Kassel, and S. Seneff, "Speech database development: design and analysis of the acoustic-phonetic corpus," in *Proc. DARPA Speech Recognition Workshop*. Feb. 1986, pp. 100–109, Report No. SAIC-86/1546.

[25] L. Deng and J. Chen, "sequence classification using the high-level features extracted from deep neural networks," in *Proc. ICASSP*, 2014.

[26] 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.