Ensemble Deep Learning for Speech Recognition

Li Deng and John C. Platt

Microsoft Research, One Microsoft Way, Redmond, WA, USA

deng@microsoft.com; jplatt@microsoft.com

Abstract

Deep learning systems have dramatically improved the accuracy of speech recognition, and various deep architectures and learning methods have been developed with distinct strengths and weaknesses in recent years. How can ensemble learning be applied to these varying deep learning systems to achieve greater recognition accuracy is the focus of this paper. We develop and report linear and log-linear stacking methods for ensemble learning with applications specifically to speech-class posterior probabilities as computed by the convolutional, recurrent, and fully-connected deep neural networks. Convex optimization problems are formulated and solved, with analytical formulas derived for training the ensemble-learning parameters. Experimental results demonstrate a significant increase in phone recognition accuracy after stacking the deep learning subsystems that use different mechanisms for computing high-level, hierarchical features from the raw acoustic signals in speech.

Index Terms: speech recognition, deep learning, ensemble learning, log-linear system combination, stacking

1. Introduction

The success of deep learning in speech recognition started with the fully-connected deep neural network (DNN) [24][40][7][18][19][29][32][33][10][11][39]. As reviewed in [16] and [11], during the past few years the DNN-based systems have been demonstrated by four major research groups in speech recognition to provide significantly higher accuracy in continuous phone and word recognition both than the earlier state-of-the-art GMM-based systems [1] and than the earlier shallow neural network systems [5][26]. More recent advances in deep learning techniques as applied to speech include the use of locally-connected or convolutional deep neural networks (CNN) [30][9][10][31] and of temporally (deep) recurrent versions of neural networks (RNN) [14][15][8][6], also considerably outperforming the early neural networks with convolution in time [36] and the early RNN [28].

While very different phone recognition error patterns (not just the absolute error rates) were found between the GMM and DNN-based systems that helped ignite the recent interest in DNNs [12], we found somewhat less dramatic differences in the error patterns produced by the DNN, CNN, and RNN-based speech recognition systems. Nevertheless, the differences are pronounced enough to warrant ensemble learning to combine these various systems. In this paper, a method is described that integrates the posterior probabilities produced by different deep learning systems using the framework of stacking as a class of techniques for forming combinations of different predictors to give improved prediction accuracy [37][4][13].

Taking a simplest yet rigorous approach, we use linear predictor combinations in the same spirit as stacked regressions of [4]. In our formulation for the specific problem at hand, each predictor is the frame-level output vectors of either the DNN, CNN, or RNN subsystem given the common filterbank acoustic feature sequences as the subsystems' inputs. However, our approach presented in this paper differs from that of [4] in several aspects. We learn the stacking parameters without imposing non-negativity constraints, as we have observed from experiments that such constraints are often automatically satisfied (see Figure 2 and related discussions in Section 4.3). This advantage derives naturally from our use of full training data, instead of just cross-validation data as in [4], to learn the stacking parameters while using the cross-validation data to tune the hyper-parameters of stacking for ensemble learning. We treat the outputs from individual deep learning subsystems as fixed, high-level hierarchical "features" [8], justifying re-use of the training data for learning ensemble parameters after learning the low-level CNN, CNN, and RNN subsystems' weight parameters. In addition to linear stacking proposed in [4], we also explore log-linear stacking in ensemble learning, which has some special computational advantage only in the context of softmax output layers in our deep learning subsystems. Finally, we apply stacking only at a component of the full ensemble-learning system --- at the frame level of acoustic sequences. Following the rather standard DNN-HMM architecture adopted in [40][7][24], the CNN-HMM in [30][10][31], and the RNN-HMM in [15][8][28], we perform stacking at the most straightforward frame level and then feed the combined output to a separate HMM decoder. Stacking at the full-sequence level is considerably more complex and will not be discussed in this paper.

This paper is organized as follows: In Section 2, we present the setting of stacking method for ensemble learning in the original linear domain of the output layers in all deep learning subsystems. The learning algorithm based on ridge regression is derived. The same setting and a similar learning algorithm are described in Section 3, except the stacking method is changed to deal with logarithmic values of the output layers in the deep learning subsystems. The main motivation is to save the significant softmax computation, and, to this end, additional bias vectors need to be introduced as part of the ensemble learning parameters. In the experimental Section 4, we demonstrate the effectiveness of both linear and log-linear stacking methods for ensemble learning, and analyze how various deep learning mechanisms for computing high-level features from the raw acoustic signals in speech naturally give different degrees of effectiveness as measured by recognition accuracy. Finally, we draw conclusions in Section 5.

2. Linear Ensemble

To simplify the stacking procedure for ensemble-learning, we perform the linear combination of the original speech-class posterior probabilities produced by deep learning subsystems at the frame level here. A set of parameters in the form of full matrices are associated with the linear combination, which are learned using the training data consisting of the frame-level posterior probabilities of the different subsystems and of the corresponding frame-level target values of speech classes. In the testing phase, the learned parameters are used to linearly combine the posterior probabilities from different systems at the frame level, which are subsequently fed to a parameter-free HMM to carry out a dynamic programming procedure together with an already trained language model. This gives the decoded results and the associated accuracy for continuous phone or word recognition.

The combination of the speech-class posterior probabilities can be carried out in either a linear or a log-linear manner. For the former, as the topic of this section, a linear combination is applied directly to the posterior probabilities produced by different deep learning systems. For the log-linear case, which is the topic of Section 3, the linear combination (including the bias vector) is applied after logarithmic transformation on the posterior probabilities.

2.1. The setting

We use two subsystems as the example to derive linear stacking for ensemble learning without loss of generality. The method can be extended to any number of subsystems in a straightforward manner.

Let $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_i, \cdots, \boldsymbol{y}_N]$ denote the output of deep-learning subsystem-one in terms of the frame-level posterior probabilities of C classes and with a total of N frames in the data (test or training); i.e. $\boldsymbol{Y} \in R^{C\times N}$. Likewise, for subsystem-two, we have $\boldsymbol{Z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_i, \cdots, \boldsymbol{z}_N] \in R^{C\times N}$. (In our experiments to be described in Section 4, we have *N≈1.12 million* for approximately 3 hours of speech training data, and *C=183.)*

With linear ensemble learning, we generate the combined system's output at each frame *i=1, 2, ⋯, N* to be

$$\boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i \in R^C \tag{1}$$

a sequence of which, with *i=1, 2, ⋯, N*, is fed to a separate HMM to produce the phone or word sequences during testing.

The two matrices, $\boldsymbol{V} \in R^{C\times C}$ and $\boldsymbol{W} \in R^{C\times C}$, are the free parameters to be learned during training that we describe below.

2.2. Parameter estimation

To learn $\boldsymbol{V}$ and $\boldsymbol{W}$, we use the supervised learning setting, where the supervision signal is the pre-labeled speech-class targets at the frame level in the training data:

$$\boldsymbol{T} = [\boldsymbol{t}_1, \cdots, \boldsymbol{t}_i, \cdots, \boldsymbol{t}_N] \in R^{C\times N}$$

The input training data consist of posterior probabilities $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_i, \cdots, \boldsymbol{y}_N]$ and $\boldsymbol{Z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_i, \cdots, \boldsymbol{z}_N]$, where N is the total number of frames in the training set.

The loss function we adopt is the total square error. Using $\boldsymbol{L}_2$ regularization, we obtain the training objective function of

$$E = \frac{1}{2}\sum_i \| \boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i - \boldsymbol{t}_i \|^2 + \lambda_1 \| \boldsymbol{V} \|^2 + \lambda_2 \| \boldsymbol{W} \|^2, \tag{2}$$

where λ_1 and λ_2 are Lagrange multipliers, which we treat as two hyper-parameters tuned by the dev or validation data in our experiments (Section 4). Optimizing (2) by setting

$$\frac{\partial E}{\partial \boldsymbol{V}} = \boldsymbol{0} \text{ and } \frac{\partial E}{\partial \boldsymbol{W}} = \boldsymbol{0},$$

we obtain

$$\sum_i (\boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i - \boldsymbol{t}_i)\boldsymbol{y}_i^{\boldsymbol{T}} + \lambda_1 \boldsymbol{V} = \boldsymbol{0} \tag{3}$$

$$\sum_i (\boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i - \boldsymbol{t}_i)\boldsymbol{z}_i^{\boldsymbol{T}} + \lambda_2 \boldsymbol{W} = \boldsymbol{0}\ . \tag{4}$$

This set of equations can be easily simplified to

$$\boldsymbol{V}(\boldsymbol{Y}\boldsymbol{Y}^{\boldsymbol{T}} + \lambda_1 \boldsymbol{I}) + \boldsymbol{W}(\boldsymbol{Z}\boldsymbol{Y}^{\boldsymbol{T}}) = \boldsymbol{T}\boldsymbol{Y}^{\boldsymbol{T}} \tag{5}$$

$$\boldsymbol{V}(\boldsymbol{Y}\boldsymbol{Z}^{\boldsymbol{T}}) + \boldsymbol{W}(\boldsymbol{Z}\boldsymbol{Z}^{\boldsymbol{T}} + \lambda_2 \boldsymbol{I}) = \boldsymbol{T}\boldsymbol{Z}^{\boldsymbol{T}} \tag{6}$$

And the solution for learning has an analytical form:

$$[\boldsymbol{V}, \boldsymbol{W}] = [\,\boldsymbol{T}\boldsymbol{Y}^{\boldsymbol{T}}, \boldsymbol{T}\boldsymbol{Z}^{\boldsymbol{T}}] \begin{bmatrix} \boldsymbol{Y}\boldsymbol{Y}^{\boldsymbol{T}} + \lambda_1 \boldsymbol{I} & \boldsymbol{Z}\boldsymbol{Y}^{\boldsymbol{T}} \\ \boldsymbol{Y}\boldsymbol{Z}^{\boldsymbol{T}} & \boldsymbol{Z}\boldsymbol{Z}^{\boldsymbol{T}} + \lambda_2 \boldsymbol{I} \end{bmatrix}^{-\boldsymbol{1}} \tag{7}$$

3. Log-Linear Ensemble

An alternative to the linear-domain system combination is the log-linear ensemble, where Eq. (1) is modified to

$$\boldsymbol{V}\log \boldsymbol{y}_i + \boldsymbol{W}\log \boldsymbol{z}_i + \boldsymbol{b} \in R^C \tag{8}$$

where $\boldsymbol{b}$ is the bias vector as a new set of free parameters.

With this modification, Eqs. (5) and (6) are changed and expanded to

$$\boldsymbol{V}(\boldsymbol{Y}'\boldsymbol{Y}'^{\boldsymbol{T}} + \lambda_1 \boldsymbol{I}) + \boldsymbol{W}(\boldsymbol{Z}'\boldsymbol{Y}'^{\boldsymbol{T}}) + \boldsymbol{b}\sum_i \boldsymbol{y}_i^{\boldsymbol{T}} = \boldsymbol{T}'\boldsymbol{Y}'^{\boldsymbol{T}}$$

$$\boldsymbol{V}(\boldsymbol{Y}'\boldsymbol{Z}'^{\boldsymbol{T}}) + \boldsymbol{W}(\boldsymbol{Z}'\boldsymbol{Z}'^{\boldsymbol{T}} + \lambda_2 \boldsymbol{I}) + \boldsymbol{b}\sum_i \boldsymbol{z}'^{\boldsymbol{T}}_i = \boldsymbol{T}'\boldsymbol{Z}^{\boldsymbol{T}}$$

$$\boldsymbol{V}\sum_i \boldsymbol{y}'^{\boldsymbol{T}}_i + \boldsymbol{W}\sum_i \boldsymbol{z}'^{\boldsymbol{T}}_i + N\boldsymbol{b} = \sum_i \boldsymbol{t}'_i$$

where $\boldsymbol{Y}' = \log \boldsymbol{Y}$ and $\boldsymbol{Z}' = \log \boldsymbol{Z}$.

Correspondingly, the solution of Eq. (7) is changed to:

$$[\boldsymbol{V}, \boldsymbol{W}, \boldsymbol{b}] =$$

$$[\,\boldsymbol{T}'\boldsymbol{Y}'^{\boldsymbol{T}}, \boldsymbol{T}'\boldsymbol{Z}'^{\boldsymbol{T}}, \sum_i \boldsymbol{t}'_i\,] \begin{bmatrix} \boldsymbol{Y}'\boldsymbol{Y}'^{\boldsymbol{T}} + \lambda_1 \boldsymbol{I} & \boldsymbol{Z}'\boldsymbol{Y}'^{\boldsymbol{T}} & \sum_i \boldsymbol{y}_i^{\boldsymbol{T}} \\ \boldsymbol{Y}'\boldsymbol{Z}'^{\boldsymbol{T}} & \boldsymbol{Z}'\boldsymbol{Z}'^{\boldsymbol{T}} + \lambda_2 \boldsymbol{I} & \sum_i \boldsymbol{z}'^{\boldsymbol{T}}_i \\ \sum_i \boldsymbol{y}'^{\boldsymbol{T}}_i & \sum_i \boldsymbol{z}'^{\boldsymbol{T}}_i & N \end{bmatrix}^{-\boldsymbol{1}}$$

The main reason why bias vector $\boldsymbol{b}$ is introduced here for system combination in the logarithmic domain and not in the linear domain described in the preceding section is as follows. In all the deep learning systems (DNN, RNN, CNN, etc.) subject to ensemble learning in our experiments, the output layers $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are softmax. Therefore, $\boldsymbol{Y}' = \log \boldsymbol{Y}$ and $\boldsymbol{Z}' = \log \boldsymbol{Z}$ are simply the exponent of the numerator minus a constant (logarithm of the partition function), which can be absorbed by the trainable bias vector $\boldsymbol{b}$ in Eq. (8). This makes the computation of the output layers of DNN, RNN, and CNN feature extractors more efficient as there is no need to compute the partition function in the softmax. This also avoids possible numerical instability in computing softmax as we no longer need to compute the exponential in its numerator.

4. Experiments

4.1. Experimental setup

To evaluate the effectiveness of the ensemble deep learning systems described in Sections 2 and 3, we use the standard TIMIT phone recognition task as adopted in many deep learning experiments [25][24][35][9][14][15]. The regular 462-speaker training set is used. A separate dev or validation set from 50 speakers' acoustic and target data is used for tuning all hyper parameters. Results on both the individual deep learning systems and their ensembles are reported using the 24-speaker core test set, which has no overlap with the dev set.

To establish the individual deep learning systems (DNN, RNN, and CNN), we use basic signal processing methods to processing raw speech waveforms. First, short-time Fourier transform with a 25-ms Hamming window and with a fixed 10-ms frame rate is used. Then, speech feature vectors are generated using filterbank analysis. This produces 41 coefficients distributed on a Mel scale (including the energy value), along with their first and second temporal derivatives to feed to the DNN and CNN systems. The RNN system takes the input from the top hidden layer of the DNN system as its feature extractor as described in [8], and is then trained using a primal-dual method in optimization as described in [6].

In all our experiments, we set $C=183$, which is the total number of target class labels, consisting of three states for each of 61 phone-like units. After decoding, the original 61 context-independent phone-like classes are mapped to a set of 39 classes for final scoring according to the standard evaluation protocol. In our experiments, a bi-gram language model over phones, estimated from the training set, is used in decoding using dynamic programming or a monophone HMM. The language model weight is set to one, and insertion penalty is set to zero in all experiments with no tuning.

To obtain the frame-level targets required both in training the individual deep learning systems and in training the ensemble parameters, a high-quality tri-phone HMM model is trained on the training data set, which is then used to generate state-level labels based on the HMM-forced alignment.

4.2. An experimental ensemble-learning paradigm

To facilitate the description of the experiments and the interpretation of the results, we use Figure 1 to illustrate the paradigm of linear stacking for ensemble learning in the linear domain (Section 2) with two individual subsystems.

The first one is the RNN subsystem described in [8], whose inputs are derived from the posterior probabilities of speech classes (given the acoustic features $\boldsymbol{X}$), denoted by $\mathrm{P}_{\mathrm{DNN}}(C|\boldsymbol{X})$ as computed from a separate DNN subsystem shown at the lower-left corner. (The output of this DNN has also been used in stacking, not shown in Figure 1.) Since the RNN makes use of the DNN features as its input, the RNN's and DNN's outputs are expected to be strongly correlated. Our experimental results presented in Section 4.3 confirm this by showing very little accuracy improvement by stacking RNN's and DNN's outputs.

The second subsystem is one of several types of the deep-CNN described in [9], receiving inputs directly from $\boldsymbol{X}$. The network actually used in the experiments has one convolutional layer, and one homogeneous-pooling layer, which is much earlier to tune than heterogeneous pooling of [9]. On top of the pooling layer, we use two fully-connected DNN layers, followed by a softmax layer producing the output as denoted by $\mathrm{P}_{\mathrm{CNN}}(C|\boldsymbol{X})$ in Figure 1.

4.3. Experimental results

As the individual deep learning systems, we have used three types of them in our experiments: the DNN, CNN, and the RNN using the DNN derived features. The systems have been developed in our earlier work reported in [9][6][8][25]. Phone recognition accuracy for these individual systems are shown at the first three rows of Table 1. The remaining rows of Table 1 show phone recognition accuracy of various system combinations using both linear ensemble described in Section 2 and log-linear ensemble described in Section 3.

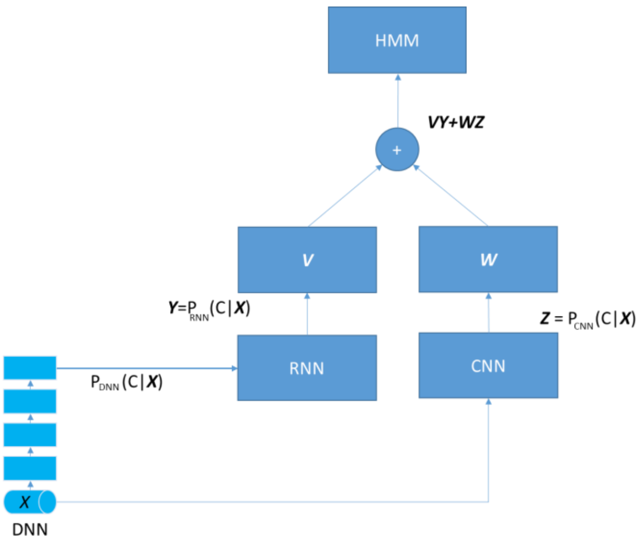

Figure 1: Illustration of linear stacking for ensemble learning between the RNN (using DNN-derived features) and the CNN subsystems.

Individual & ensemble systems	Accuracy
DNN (with 5 hidden layers)	77.9%
CNN (with homogeneous pooling)	80.5%
RNN (with DNN-derived features)	80.7%
Linear Ensemble (DNN, RNN)	80.8%
Linear Ensemble (DNN, CNN)	80.9%
Linear Ensemble (CNN, RNN)	81.6%
Log-Linear Ensemble (CNN, RNN)	81.5%
Linear Ensemble (DNN, CNN, RNN)	81.7%
Log-Linear Ensemble (DNN, CNN, RNN)	81.7%

Table 1: Phone recognition accuracy after HMM decoding with three individual deep learning subsystems and their linear or log-linear stacking for ensemble learning at the frame level.

Note all ensembles improve recognition accuracy but to a varying degree. Importantly, we see a clear pattern regarding what kinds of improvement result from what types of system combinations. First, the ensemble between the DNN and RNN gives very small improvement over the RNN (from 80.7% to 80.8%). This can be easily understood since the RNN takes the input directly from the output of the DNN, as shown in Figure 1, and hence there is huge information overlap. Second, likely due to the less information overlap, an ensemble between the DNN and CNN gives a more substantial accuracy gain over the better of the two (i.e., CNN, from 80.5% to 80.9%). Third, an ensemble between the CNN and RNN gives the greatest improvement over both individual subsystems which are comparable in accuracy but with much less information overlap. Further, we found that liner and log-linear ensembles produce very similar accuracy but the latter is more efficient in run-time computation. This is because, as pointed out in Section 3, the log-linear stacking avoids the computation of full softmax functions, which is quite substantial for large vocabulary speech systems [33][19].

In Figure 2, we provide a typical example of the estimated matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ used in linear stacking between the RNN and CNN subsystems. These estimates are unique, since the optimization problem is formulated to be convex and we have derived the analytical solution. As shown, both $\boldsymbol{V}$ and $\boldsymbol{W}$ are close to diagonal, but the magnitudes of the diagonal elements vary rather widely, far from being constant. In fact, in the initial exploration before the current study, we used two scalar weights, i.e., $a\boldsymbol{y}_i + b\boldsymbol{z}_i$ instead of $\boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i$ in Eq. (1), to carry out system combinations, amounting to forcing $\boldsymbol{V}$ and $\boldsymbol{W}$ to be diagonal with identical diagonal elements. Accuracy improvement observed using scalar weights with cross validation motivated the formal study reported in this paper with more general and learnable matrices and with greater gains achieved.

5. Discussion and Conclusion

Deep learning has gone a long way from the early small experiments [17][12][13][25][24][2] using the fully-connected DNN to increasingly larger experiments [16][39][11][29][33][18] and to the exploitation of more advanced deep architectures such as the deep CNN [30][9][31] which takes into account invariant properties of speech spectra trading off with speech-class confusion and RNN [20][21][22][23][27][34][14][15] which exploits dynamic properties of speech. While the recent work already started to integrate strengths of different sorts of deep models in a primitive manner --- e.g., feeding outputs of the DNN as high-level features to an RNN [6][8], the study reported in this paper gives a more principled way of performing ensemble learning based on a well-established paradigm of stacking [4]. Our work goes beyond the basic stacked linear regression method of [4] by extending it to the log-linear case and by adopting a more effective learning scheme. In our method designed for deep learning ensembles, the stacking parameters are learned using both training and validation data computed from the outputs of different deep networks (DNN, CNN, and RNN) serving as high-level features. In this way, non-negativity constraints as required by [4] are no longer needed.

A more principled way of ensemble learning by stacking is to perform it at the full-sequence level, rather than at the frame level as attempted in this paper. The greater complexity of this pursuit leaves it to our future work. Also, with an increasingly large amount of training data and computing power, it is conceivable that the stacking parameters and those in all the constituent deep networks can be learned jointly using the unified backpropagation in an end-to-end fashion.

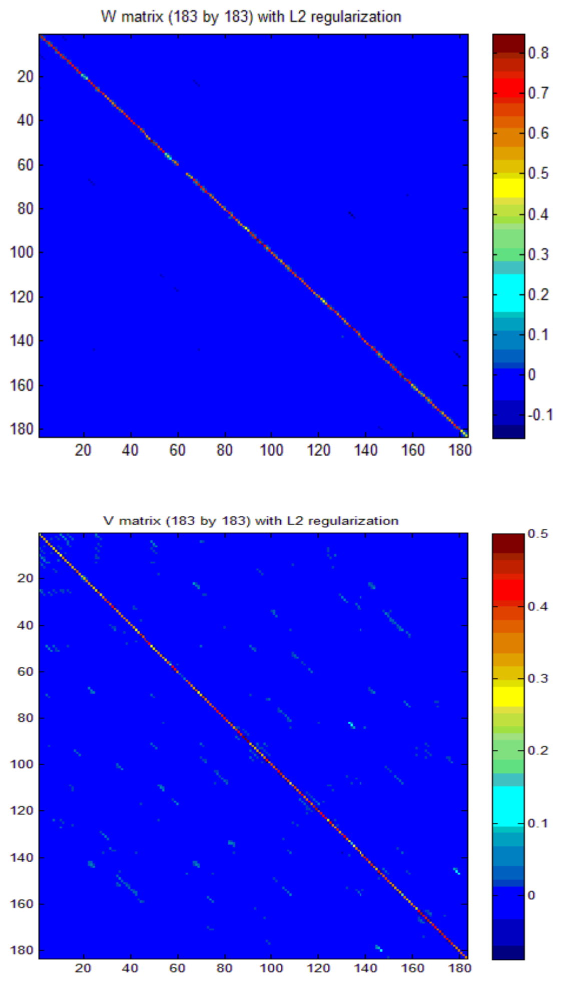

Figure 2: Examples of the learned matrices, both 183 by 183 in size, produced by linear stacking for assemble learning.

6. Acknowledgements

We thank Jianshu Chen of UCLA, who worked with us as an intern during the summer of 2013 on the learning method published recently in [6]. This method was used to train the RNN system of [8], which was in turn used in our stacking experiments described in this paper.

7. References

[1] Baker, J., Deng, L., Glass, J., Khudanpur, S., Lee, C.-H., Morgan, N., and O'Shaughnessy, D. "Research developments and directions in speech recognition and understanding," IEEE Sig. Proc. Mag., vol. 26, no. 3, May 2009, pp. 75-80.

[2] Bengio, Y., Courville, A., and Vincent, P. “Representation learning: A review and new perspectives,” IEEE Trans. PAMI, vol. 38, pp. 1798-1828, 2013.

[3] Bengio, Y., Boulanger, N., and Pascanu, R. “Advances in optimizing recurrent networks,” Proc. ICASSP, 2013.

[4] Breiman, L. “Stacked regression,” Machine Learning, Vol. 24, pp. 49-64, 1996.

[5] Bourlard, H. and Morgan, N., Connectionist Speech Recognition: A Hybrid Approach, Kluwer, 1993.

[6] Chen, J. and Deng, L. “A primal-dual method for training recurrent neural networks constrained by the echo-state property,” Proc. Int. Conf. Learning Representations, April, 2014.

[7] Dahl, G., Yu, D., Deng, L., and Acero, A. “Context-dependent, pre-trained deep neural networks for large vocabulary speech recognition,” IEEE Trans. Audio, Speech, & Language Proc., Vol. 20, pp. 30-42, 2012.

[8] Deng, L. and Chen, J. “Sequence classification using the high-level features extracted from deep neural networks,” Proc. ICASSP, 2014.

[9] Deng, L., Abdel-Hamid, O., and Yu, D. “A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion,” Proc. ICASSP, 2013.

[10] Deng, L., Li, J., Huang, K., Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero. “Recent advances in deep learning for speech research at Microsoft,” Proc. ICASSP, 2013.

[11] Deng, L., Hinton, G., and Kingsbury, B. “New types of deep neural network learning for speech recognition and related applications: An overview,” Proc. ICASSP, 2013.

[12] Deng, L., Yu, D., and Hinton, G. “Deep Learning for Speech Recognition and Related Applications” NIPS Workshop, 2009.

[13] Deng, L., Yu, D., and Platt, J. “Scalable stacking and learning for building deep architectures,” Proc. ICASSP, 2012.

[14] Graves, A., Mohamed, A., and Hinton, G. “Speech recognition with deep recurrent neural networks,” Proc. ICASSP, 2013.

[15] Graves, A., Jaitly, N., and Mohamed, A. “Hybrid speech recognition with deep bidirectional LSTM,” Proc. ASRU, 2013.

[16] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B., “Deep Neural Networks for Acoustic Modeling in Speech Recognition,” IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, 2012.

[17] Hinton, G. and Salakhutdinov, R. “Reducing the dimensionality of data with neural networks,” Science, vol. 313. no. 5786, pp. 504 - 507, July 2006.

[18] Jaitly, N., Nguyen, P., and Vanhoucke, V. “Application of pre-trained deep neural networks to large vocabulary speech recognition,” Proc. Interspeech, 2012.

[19] Kingsbury, B., Sainath, T., and Soltau, H. “Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization,” Proc. Interspeech, 2012.

[20] Maas, A., Le, Q., O'Neil, T., Vinyals, O., Nguyen, P., and Ng, P. “Recurrent neural networks for noise reduction in robust ASR,” Proc. Interspeech, 2012.

[21] Martens, J. and Sutskever, I. “Learning recurrent neural networks with Hessian-free optimization,” Proc. ICML, 2011.

[22] Mikolov, T., Deoras, A., Povey, D., Burget, L., and Cernocky, J. “Strategies for training large scale neural network language models,” Proc. ASRU, 2011.

[23] Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., and Khudanpur, S. “Recurrent neural network based language model,” Proc. ICASSP, 2010, 1045–1048.

[24] Mohamed, A., Dahl, G. and Hinton, G. “Acoustic modeling using deep belief networks”, IEEE Trans. Audio, Speech, and Language Proc. Vol. 20., January 2012.

[25] Mohamed, A., Yu, D., and Deng, L. “Investigation of full-sequence training of deep belief networks for speech recognition,” Proc. Interspeech, 2010.

[26] Morgan, N. “Deep and wide: Multiple layers in automatic speech recognition,” IEEE Trans. Audio, Speech, and Language Processing, Vol. 20 (1), January 2012.

[27] Pascanu, R., Mikolov, T., and Bengio, Y. “On the difficulty of training recurrent neural networks,” Proc. ICML, 2013.

[28] Robinson, A. “An application of recurrent nets to phone probability estimation,” IEEE Trans. Neural Networks, Vol. 5, pp. 298-305, 1994.

[29] Sainath, T., Kingsbury, B., Soltau, H., and Ramabhadran, B. “Optimization Techniques to Improve Training Speed of Deep Neural Networks for Large Speech Tasks,” IEEE Trans. Audio, Speech, and Language Processing, vol.21, no.11, pp.2267-2276, Nov. 2013.

[30] Sainath, T., Mohamed, A., Kingsbury, B., and Ramabhadran, B. “Convolutional neural networks for LVCSR,” Proc. ICASSP, 2013.

[31] Sainath, T., Kingsbury, Mohamed, A., Dahl, G., Saon, G., Soltau, H., Beran, T., Aravkin, A., and B. Ramabhadran. “Improvements to deep convolutional neural networks for LVCSR,” Proc. ASRU, 2013.

[32] Sainath, T., Kingsbury, B., Ramabhadran, B., Novak, P., and Mohamed, A. “Making deep belief networks effective for large vocabulary continuous speech recognition,” Proc. ASRU, 2011.

[33] Seide, F., Li, G., and Yu, D. “Conversational speech transcription using context-dependent deep neural networks,” Proc. Interspeech, 2011.

[34] Sutskever, I., Martens J., and Hinton, G. “Generating text with recurrent neural networks,” Proc. ICML, 2011.

[35] Triefenbach, F., Jalalvand, A., Demuynck, K., Martens, J.-P. “Acoustic modeling with hierarchical reservoirs,” IEEE Trans. Audio, Speech, and Language Processing, vol.21, no.11, pp. 2439-2450, Nov. 2013.

[36] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. “Phoneme recognition using time-delay neural networks,” IEEE Trans. Acoust. Speech, and Signal Proc., vol. 37, pp. 328-339, 1989.

[37] Wolpert, D. “Stacked generalization,” *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[38] Yao, K., Zweig, G., Hwang, M., Shi, Y., and Yu, D. “Recurrent Neural Networks for Language Understanding,” Proc. Interspeech, 2013.

[39] Yu, D., Deng, L., and Seide, F. “The deep tensor neural network with applications to large vocabulary speech recognition,” IEEE Trans. Audio, Speech, and Language Processing, vol. 21, no. 2, pp. 388-396, 2013.

[40] Yu, D., Deng, L., and Dahl, G.E., “Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition,” NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2010.