Rectified Linear Neural Networks with Tied-Scalar Regularization for LVCSR

Shiliang Zhang[1], *Hui Jiang*[2], *Si Wei*[1], *Li-Rong Dai*[1]

[1]National Engineering Laboratory for Speech and Language Information Processing
University of Science and Technology of China, Hefei, Anhui, P. R. China
[2] Department of Electrical Engineering and Computer Science
York University, 4700 Keele Street, Toronto, Ontario, M3J 1P3, Canada
zsl2008@mail.ustc.edu.cn, hj@cse.yorku.ca, siwei@iflytek.com, lrdai@ustc.edu.cn

Abstract

It is known that rectified linear deep neural networks (RL-DNNs) can consistently outperform the conventional pre-trained sigmoid DNNs even with a random initialization. In this paper, we present another interesting and useful property of RL-DNNs that we can learn RL-DNNs with a very large batch size in stochastic gradient descent (SGD). Therefore, the SGD learning can be easily parallelized among multiple computing units for much better training efficiency. Moreover, we also propose a tied-scalar regularization technique to make the large-batch SGD learning of RL-DNNs more stable. Experimental results on the 309-hour Switchboard (SWB) task have shown that we can train RL-DNNs using batch sizes about 100 times larger than those used in the previous work, thus the learning of RL-DNNs can be accelerated by over 10 times when 8 GPUs are used. More importantly, we have achieved a word error rate of 13.8% with a 6-hidden-layer RL-DNN trained by the frame-level cross-entropy criterion with the tied-scalar regularization. To our knowledge, this is the best reported performance on this task under the same experimental settings.

Index Terms: rectified linear units, RL-DNN, LVCSR, tied-scalar regularization

1. Introduction

Recently, neural networks have revived as a popular model in machine learning under the name of deep learning. Deep learning aims at learning neural networks with very deep architecture, such as deep neural networks (DNNs), which significantly outperform other machine learning methods and yield the state of the art performance in many real-world applications, such as speech recognition, computer vision and many others.

As the basic building block of DNNs, each neuron (hidden node) imposes a nonlinear activation function from its input to output. Traditionally, the standard logistic sigmoid and hyperbolic tangent functions are widely used in the neural networks literature. More recently, in [1, 2, 3], it has proposed to use a simpler nonlinear neuron, namely rectified linear unit (ReLU), which takes a rectified linear function as its activation function from input to output. In many large scale real-world applications [4, 5, 6, 7, 8], ReLU based DNNs (RL-DNN) have demonstrated several major advantages over the traditional DNNs using logistic sigmoid or hyperbolic functions [5]. Firstly, RL-DNNs normally yield better recognition performance than the regular sigmoid DNNs. Secondly, the training speed of RL-DNNs is much faster than that of the regular sigmoid DNNs because the learning process of RL-DNNs seems to converge faster. Next, rectified linear units generate exact zeros when inputs are not aligned with the internal weights in the model, as opposed to the sigmoid units that produce small noisy values. As a result, a learned RL-DNN is much sparser than its counterpart using logistic sigmoid units. The increased sparsity is believed to improve model generalization [3]. Moreover, based on the experimental observations in [4, 5], it has been widely conjectured that RL-DNNs are much easier to learn since the piece-wise linear property arising from ReLUs may be beneficial from the optimization perspectives.

As we know, the learning objective functions of DNNs are always highly non-convex, where many distinct local minima co-exist in the model parameter space. Therefore, the stochastic gradient descent (SGD) algorithm [9, 10, 11] is usually used to optimize the non-convex objective function to learn DNNs, which is believed to help the learning to escape from poor local optima. Unfortunately, using small mini-batches in SGD becomes a major computation bottleneck to parallelize the DNN learning among multiple GPUs or to distribute the learning to large CPU clusters.

In this paper, we have observed an interesting and useful property of RL-DNNs that we can learn RL-DNNs reliably using a very large mini-batch size, even up to 100k (102400), which is about 100 times larger than those used in the previous work. Therefore, it can largely overcome the major hurdle in the parallel training of DNNs since the learning can be easily parallelized by splitting each large "mini-batch" into multiple computing units for much faster training turnaround. As shown in our experiments on the Switchboard task, we can dramatically speedup the RL-DNN training by more than 10 times when using 8 GPUs in the parallel training. In our large mini-batch SGD training, we still need to carefully tune the learning rate as usual for the best possible performance. In this paper, we first give a heuristic but useful strategy to adjust the learning rates according to the used mini-batch size. More importantly, in order to make the large mini-batch training of RL-DNNs reliable and effective, we have proposed a new tied-scalar regularization technique, which can yield a significant performance improvement over the conventional methods. For example, we have achieved a word error rate of 13.8% on the 309-hour Switchboard (SWB) task with a 6-hidden layer RL-DNN that takes filter bank features (FBK) as input and is trained with the frame-level cross-entropy criterion, which is about 11.5% relative performance improvement over the conventional pre-trained sigmoid DNNs. To the best of our knowledge, this is the best reported perfor-

This work was partially supported by the National Nature Science Foundation of China (Grant No. 61273264) and the electronic information industry development fund of China (Grant No. 2013-472).

mance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector X as input, an $(L+1)$-layer DNN, consisting of L hidden nonlinear layers, $\ell = 1...L$, and one output layer ($\ell = L+1$), is used to model the posterior probability $\Pr(s|X)$ of each output target, s, for classification. Assume that it contains N_ℓ hidden nodes in the ℓ-th layer. In RL-DNNs, each hidden node adopts the so-called rectified linear activation function, i.e., $f(x) = \max(0, x)$, to compute from the linear activations of the current layer, $\mathbf{a}^\ell$, to its outputs, $\mathbf{h}^\ell$, which are in turn fed to the next layer as input. Assume the input is $\mathbf{h}^0 = X$, an RL-DNN works as follows:

$$\mathbf{a}^\ell = \mathbf{W}^\ell \mathbf{h}^{\ell-1} + \mathbf{b}^\ell, \quad (1 \le \ell \le L+1), \tag{1}$$

$$\mathbf{h}^\ell = f\left(\mathbf{a}^\ell\right) = \max\left(0, \mathbf{a}^\ell\right), \quad (1 \le \ell \le L), \tag{2}$$

where $\mathbf{W}^\ell$ and $\mathbf{b}^\ell$ represent the weight matrix and the bias vector for layer ℓ. For classification, the output layer uses *softmax* to compute posterior probabilities as follows:

$$y_s = h_s^{L+1} = \Pr(s|X) = \mathrm{softmax}_s\left(\mathbf{a}^{L+1}\right), \tag{3}$$

where y_s denotes the s-th element of the output vector $\mathbf{y}$.

The following cross-entropy (CE) error objective function is used to learn RL-DNNs in speech recognition:

$$\begin{aligned}\mathcal{F}(\mathbf{\Theta}) &= -\sum_{r=1}^{R} \log\left(\mathrm{softmax}_{s_r}(\mathbf{a}^{L+1})\right) \\ &= -\sum_{r=1}^{R} \log \Pr(s_r|X_r),\end{aligned} \tag{4}$$

where $\mathbf{\Theta} = \{\mathbf{W}^\ell, \mathbf{b}^\ell \mid \ell = 1, 2, \cdots, L+1\}$ denotes all connection weight matrices in DNN, and s_r denotes the target label of input feature vector X_r. Therefore, the learning of RL-DNNs can be formulated as the following general optimization problem:

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta} \in \mathbb{R}^{\mathbf{N}}} \mathcal{F}(\mathbf{\Theta}). \tag{5}$$

The above optimization problem is generally regarded as being difficult to solve since it is non-convex and high-dimensional.

2.1. Learning RL-DNNs with SGD

We usually use the mini-batch SGD algorithm to solve the optimization problem in eq.(5) to learn RL-DNNs. For example, a widely used batch size is 1024 for speech recognition on many tasks. Unfortunately, it becomes difficult to achieve an effective parallelization for the DNN training with such a small mini-batch because of the communication overhead problem. The asynchronous stochastic gradient descent (ASGD) in [12] is used to parallelize the DNN training among multiple GPUs. However, it needs to use even smaller mini-batch size in early iterations of ASGD for a good convergence, which significantly limits the parallelization speedup and can not take advantage of the full computing sources in each GPU.

In this work, we have found that RL-DNNs can be learned with a very large batch size. We have evaluated the performance of RL-DNNs trained using various batch sizes, such as 1k (1024), 4k (4096), 10k (10240), 20k (20480), 50k (51200) and 100k (102400). In these experiments, we may have to adjust the learning schedule accordingly for different mini-batch sizes. The rule of thumb is that a small learning rate is used for a small mini-batch size while a larger learning rate for a large batch size. This is due to the fact that the number of model updates per epoch is inversely proportional to the mini-batch size so that we need to increase the learning rates for larger mini-batch sizes to ensure comparable model updates per epoch. Based on our experiments, we have found that the initial learning rate should be proportional to the used mini-batch size as follows:

$$\frac{lr_1}{lr_2} = \frac{\mathrm{minibatchSize}_1}{\mathrm{minibatchSize}_2} \tag{6}$$

Based on our previous work, when we use a normal mini-batch size of 1k, we use an initial learning rate of 0.02. For other larger mini-batch sizes, we can use eq.(6) to proportionally set the initial learning rates. For example, we should set the initial learning rate to 0.4 for the mini-batch size of 20k. After the initial learning rate is set, the learning rate is kept fixed as long as the frame classification accuracy on a cross-validation (CV) set improves by at least 0.5%. After that, we continue six more epochs of the SGD training, where the learning rate is halved after each epoch.

Another important trick to train RL-DNNs is that we must use a much smaller initial weights. In our experiments, all weights are initialized based on the normalized uniform random initialization in [13]. As the rectifier linear function is unbounded, the activations of hidden units might grow without limit. To handle this potential numerical problem, we adopt a slight modification to the normalized initialization method [13] by multiplying a constant factor to control the dynamic range of the initial weights as follows:

$$W \sim [-\beta \cdot \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \beta \cdot \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}] \tag{7}$$

where n_i is the number of units in the i-th layer and β is set to 0.5 in all of our experiments.

3. RL-DNN with Tied-Scalar Regularization

In section 2, we have shown that RL-DNNs can be trained by SGD with a very large mini-batch size. In [14], it mentions that each update of model weights should be about 10^{-3} of the weights in order of magnitude. When we use larger learning rates for the larger mini-batch sizes in SGD, we need to carefully tune the learning rate as described in section 2.1. In order to make the learning process more reliable, we further propose a new tied-scalar regularization method, which is found to be particularly useful for learning RL-DNNs with larger mini-batch sizes. Our tied-scalar idea is inspired by the L_2 norm regularization used in [15], which makes it possible to learn DNNs with a very large initial learning rate. As for L_2 norm regularization, it uses a preset upper bound to limit the L_2 norm of all fan-in weight vectors in each hidden layer. After each weight update, we renormalize the norm of each weight vector to ensure it is not out of the bound. Unfortunately, the L_2 norms of DNN weight vectors vary significantly in the dynamic range from one data set to another. For every new task, we may need to conduct many experiments to manually search for the suitable L_2 norm bound. Therefore, the main idea of our proposed tied-scalar regularization is to automatically learn the

L_2 norm bound from data. In the tied-scalar regularization, we constrain the L_2 norm of each fan-in weight vector to each individual hidden node of DNNs to be less than 1, $\|\mathbf{w}_k^\ell\| \leq 1$ ($\mathbf{w}_k^\ell$ denotes k-th row vector of the weight matrix $\mathbf{W}^\ell$ in the ℓ-th hidden layer). And all weight vectors in each hidden layer share a common scalar, $\alpha^\ell > 0$, which is used to adjust weight vectors to proper lengths. The tied-scalars are all learned in the back-propagation in the same way as other weights. Since α^ℓ is tied to many different weight vectors in DNNs, as we observed in our experiments, it never diverges in the learning and always converges to a reasonable number.

In other words, the proposed RL-DNNs with tied-scalar regularization can be expressed as the following form:

$$\mathbf{a}^\ell = \alpha^\ell \mathbf{W}^\ell \mathbf{h}^{\ell-1} + \mathbf{b}^\ell \quad (\|\mathbf{w}_k^\ell\| \leq 1 \;\; \forall k, \ell) \tag{8}$$

$$\mathbf{h}^\ell = f\left(\mathbf{a}^\ell\right) = \max\left(0, \mathbf{a}^\ell\right). \tag{9}$$

We can use the chain rule to derive the derivatives for the tied-scalars, which takes the following form:

$$\frac{\partial \mathcal{F}}{\partial \alpha^\ell} = \sum_{j=1}^{N_\ell} \frac{\partial \mathcal{F}}{\partial h_j^\ell} \frac{\partial h_j^\ell}{\partial \alpha^\ell} = \sum_{j=1}^{N_\ell} e_j^\ell (\mathbf{w}_j^\ell \cdot \mathbf{h}^{\ell-1}) \tag{10}$$

where e_j^ℓ denotes for the error signal computed for the j-th hidden node in the layer ℓ, N_ℓ denotes the number of the hidden units in ℓ-th layer. As for the initialization, we use the maximum L_2 norm of all (randomly) initial weight vectors in each layer to initialize all α^ℓ, and all weight vectors in each layer are normalized by α^ℓ to satisfy the norm constraints: $\|\mathbf{w}_k^\ell\| \leq 1$ $(\forall k, \ell)$.

During the SGD learning, we need to constrain the L_2 norm of each weight vector if its norm exceeds 1 after each update:

$$\mathbf{w}_k^\ell \leftarrow \frac{\mathbf{w}_k^\ell}{\|\mathbf{w}_k^\ell\|} \quad \text{if } \|\mathbf{w}_k^\ell\| > 1 \tag{11}$$

The proposed tied-scalar regularization can prevent all DNN weights from growing too large, which makes it suitable for learning RL-DNNs with a very large initial learning rate for those larger mini-batch sizes.

4. Experiments

In this paper, we have examine how to training RL-DNNs with large mini-batch sizes and further investigate the proposed tied-scalar regularization on the Switchboard (SWB) database. The SWB training data consists of 309-hour Switchboard-I training database and 20-hour Call Home English data. We divide the whole training data into two sets: training set and cross-validation set. The training set contains 99.5% training data, and the cross-validation set contains the remaining 0.5%. Evaluation is performed in terms of word error rate (WER) on the NIST 2000 Hub5 evaluation set (containing 1831 utterances), denoted as Hub5e00.

4.1. Baseline systems (GMM/HMM and sigmoid DNNs)

The GMM/HMM baseline is a standard tied-state cross-word tri-phone system, using the 39-dimension PLP features (static, first and second derivatives) as feature and Gaussian mixture models (GMM) as acoustic model, which is estimated with the maximum likelihood estimation (MLE) and then discriminatively trained based on the minimum phone error (MPE) criterion [16]. Before the model training, all feature vectors are pre-processed with the cepstral mean and variance normalization (CMVN) per conversation side. The final hidden Markov model (HMM) consists of 8,991 tied states and 40 Gaussian components per state. In the decoding, we use a trigram language model (LM) that is trained by using 3 million words from the training transcripts and another 11 million words of the Fisher English Part 1 transcripts. The performance of the baseline GMM/HMMs systems is listed in Table 1.

Table 1: Word error rates (WER in %) of various baseline systems in Switchboard.

model	method		Hub5e00
GMM-HMM	MLE		28.7
	MPE		24.7
DNN-HMM	PLP	5*2048	16.5
		6*2048	16.2
	FBK	5*2048	16.1
		6*2048	15.6

For the baseline DNN systems, we use the logistic sigmoid function as the activation function for all hidden nodes. We follow the same training procedure as described in [17, 18, 19, 20] to train the conventional context dependent DNN/HMM with the tied-triphone state alignment obtained from the above MLE trained GMM/HMMs baseline system. The input vectors are either 39-dimension PLP or 123-dimension mel-warped filter-bank (FBK) features concatenated from all consecutive frames within a long context window of (5+1+5). The sigmoid DNN is first pre-trained using the RBM-based layer-wise pre-training and then fine-tuned with 10 epochs of frame-level cross-entropy (CE) training. In the fine-tuning, we used SGD with mini-batches of 1024 (1k) frames. The initial learning rate is set to 0.2. We have trained sigmoid DNNs with 5 and 6 hidden layers and 2,048 nodes per layer. The performance of these baseline DNN/HMMs systems is also listed in Table 1 for comparison.

4.2. RL-DNNs

We train RL-DNNs (with 5 or 6 hidden layers of 2048 ReLU nodes per layer) using the conventional back-propagation (BP) with varying batch sizes, being 1k (1024), 10k (10240), 20k (20480), 50k (51200) and 100k (102400). The learning parameters of RL-DNNs are similar to those of sigmoid DNNs. The learning schedule of RL-DNNs is described in section 2.1. Experimental results in Table 2 show that we can still achieve very competitive recognition performance when we use much larger mini-batch sizes to train RL-DNNs. For instance, the recognition performance of the 6-hidden-layer RL-DNN only changes from 15.0% and 15.5% when the mini-batch size is increased from 1k to 100k. Moreover, the learning curves in Figure 1 also indicate that the learning of RL-DNNs converges very well for a wide range of mini-batch sizes. Particularly, for a 6-hidden-layer RL-DNN trained using a batch size of 10k, we have achieved 15.0% in WER, a 1.2% absolute error reduction over the baseline pre-trained sigmoid DNNs in Table 1. Another advantage of using large batch size is that the training can be easily parallelized among multiple computing units to significantly improve training efficiency. In our experiments, we have implemented a parallel scheme to train a 6-hidden-layer RL-DNN with a mini-batch size of 8k using a system of 4 GPUs (NVIDIA 750 Tesla K20). It has shown that the training speed can be expedited by about 3 times by simply distribut-

Table 2: Word error rates (WER in %) in the Switchboard of RL-DNNs learned using PLP features with various batch sizes and training speedup factors when 8 GPUs are used in parallel.

batch size	RL-DNN			
	5*2048		6*2048	
	WER(%)	speedup	WER(%)	speedup
1k	15.9	-	15.4	-
10k	15.6	x5.4	**15.0**	x5.4
20k	**15.5**	x8.2	15.3	x8.4
50k	15.7	x11.2	15.3	x11.7
100k	16.0	x12.2	15.5	x12.7

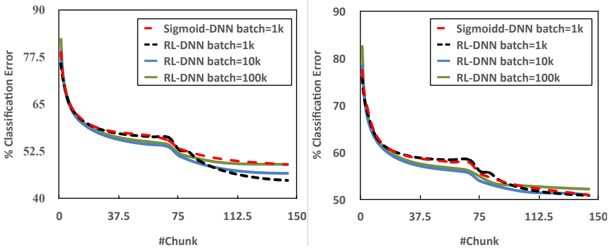

Figure 1: Comparison of various learning curves of sigmoid DNNs and RL-DNNs (6 hidden layers of 2048 neurons) on the Switchboard task using different batch sizes (1k, 10k and 100k); The left figure is for training data and the right figure for the 0.5% held-out development set.

ing the gradient computation of each mini-batch into 4 GPUs. In this case, each epoch takes about 1.6 hours, consisting of 1.3 hours of computation in 4 GPUs and 0.3 hours of communication overhead to collect gradients and redistribute the updated models among 4 GPUs. This is about 4.6 times faster than the baseline system trained with a mini-batch of 1k. The parallel training method can be easily extended to even more GPUs with even larger batch sizes. The good thing is that the larger the mini-batch size is, the less communication overhead is involved. In Table 2, we have listed all training speedup factors for each case based on a simulation environment of 8 GPUs [21, 22]. The results have shown that the learning of a 6-hidden-layer RL-DNN can be accelerated by over 11.7 times[1] when we use 8 GPUs and set the mini-batch size to 50k. In this case, we can still achieve a very good performance (15.3% in WER).

4.3. RL-DNNs with tied-scalar regularization

We further investigate the performance of RL-DNNs with the tied-scalar regularization. Here we have trained RL-DNNs using either PLP or FBK features. The training scheme of RL-DNNs with tied-scalars is similar to that of RL-DNNs except that we use a different learning rate for all tied-scalars, which is set to 0.002 in our experiments. The learning rate of the tied-scalars is kept fixed during the training. The advantage of using tied-scalars is that we can train RL-DNNs with a large batch size more reliably. We just need to set an initial learning as eq. (6). The learning curves of the tied-scalars of all hidden layers are plotted in Figure 2, which indicates that these tied-scalars gradually increase during the learning and can eventually converge very well at the end. In this experiment, we can see that

[1]For SWB experiments, the average times for one epoch of the RL-DNN training are 6.7 hours and 4.5 hours (measured by a NVIDIA Tesla K20 GPU) when we use the mini-batch size of 1k and 5k respectively.

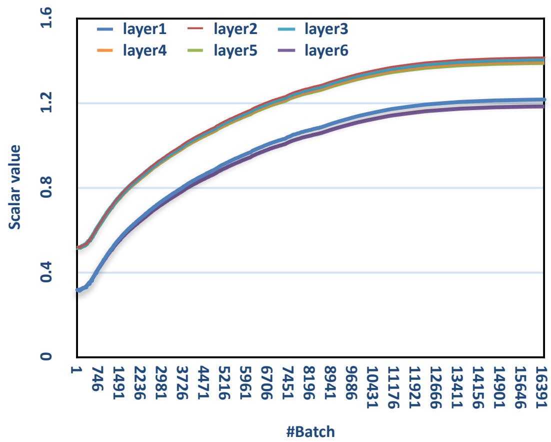

Figure 2: The learning curves of tied-scalar for a 5-hidden-layer RL-DNN with tied-scalar trained using batch size of 10k.

Table 3: Word error rates (WER in %) of RL-DNNs with tied-scalar in SWB.

Input	hidden layer	batch size		
		10k	100k	200k
PLP	5*2048	15.1	15.2	16.0
	6*2048	**14.7**	15.2	-
FBK	6*2048	**13.8**	14.1	-

the tied-scalars in all hidden layers converge to 1.4 while those in the input and output layers converge to 1.2. Compared to the L_2 norm regularization in [15], the proposed tied-scalar regularization can automatically learn the optimal upper bound for the L_2 norm. Experimental results in Table 3 show that the proposed tied-scalar regularization can further improve the performance of RL-DNNs significantly. For example, we have achieved a word error rate of 15.1% by using PLP feature for a 5-hidden-layer RL-DNN with tied-scalar while the baseline system in Table 1 is 16.5%. Moreover, for a 6-hidden-layer RL-DNN with tied-scalars (use FBK as input) trained using a batch size of 10k, we have achieved 13.8% in WER. To our knowledge, this is the best reported performance on this task for the speaker-independent training (without speaker-specific adaptation and normalization as in [23, 24]) using the frame-level cross-entropy error criterion.

5. Conclusions

In this work, we have presented an empirical study to show that RL-DNNs can be effectively learned with a very large mini-batch size and have further proposed a tied-scalar regularization method to make the learning of RL-DNNs with large batch sizes more reliable. In this way, the training of RL-DNNs can be easily parallelized using multiple GPUs for better efficiency. Our experiments on the Switchboard task have shown that we can speedup the learning by more than 10 times using 8 GPUs, meanwhile it can still yield very competitive performance. For example, we have achieved a WER of 13.8% with a 6-hidden-layer RL-DNN with tied-scalars trained using a mini-batch size of 10k, which is the best reported performance in this case.

6. References

[1] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2146–2153.

[2] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[3] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, vol. 15, 2011, pp. 315–323.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1106–1114.

[5] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean *et al.*, "On rectified linear units for speech processing." ICASSP, 2013.

[6] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Proc. ICASSP*, 2013.

[7] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013.

[8] L. Tóth, "Phone recognition with deep sparse rectifier neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6985–6989.

[9] L. Bottou, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nımes*, vol. 91, no. 8, 1991.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in neural information processing systems*, 2008, pp. 161–168.

[12] S. Zhang, C. Zhang, Z. You, R. Zheng, and B. Xu, "Asynchronous stochastic gradient descent for dnn training," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6660–6663.

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[14] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.

[15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[16] H. Jiang, "Discriminative training for automatic speech recognition: A survey," *Computer and Speech, Language*, vol. 24, no. 4, pp. 589–608, 2010.

[17] J. Pan, C. Liu, Z. Wang, Y. Hu, and H. Jiang, "Investigation of deep neural networks (DNN) for large vocabulary continuous speech recognition: Why DNN surpasses GMMs in acoustic modeling," in *Proc. of International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2012, pp. 301–305.

[18] Y. Bao, H. Jiang, C. Liu, Y. Hu, and L. Dai, "Investigation on dimensionality reduction of concatenated features with deep neural network for lvcsr systems," in *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, vol. 1. IEEE, 2012, pp. 562–566.

[19] Y. Bao, H. Jiang, L. Dai, and C. Liu, "Incoherent training of deep neural networks to de-correlate bottleneck features for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6980–6984.

[20] S. Zhang, Y. Bao, P. Zhou, H. Jiang, and L. Dai, "Improving deep neural networks for LVCSR using dropout and shrinking structure," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6849–6853.

[21] P. Zhou, C. Liu, Q. Liu, L. Dai, and H. Jiang, "A cluster-based multiple deep neural networks method for large vocabulary continuous speech recognition," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[22] P. Zhou, H. Jiang, L.-R. Dai, Y. Hu, and Q.-F. Liu, "State-clustering based multiple deep neural networks modeling approach for speech recognition," *IEEE/ACM Trans. on Audio, Speech and Language Processing*, vol. 23, no. 4, pp. 631–642, 2015.

[23] S. Xue, O. Abdel-Hamid, H. Jiang, and L. Dai, "Direct adaptation of hybrid DNN/HMM model for fast speaker adaptation in LVCSR based on speaker code," in *Proc. of IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*, 2014.

[24] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE/ACM Trans. on Audio, Speech and Language Processing*, vol. 22, no. 12, pp. 1713–1725, 2014.