# Convolutional Deep Maxout Networks for Phone Recognition

*László Tóth*

MTA-SZTE Research Group on Artificial Intelligence
Hungarian Academy of Sciences and University of Szeged, Hungary
tothl@inf.u-szeged.hu

## Abstract

Convolutional neural networks have recently been shown to outperform fully connected deep neural networks on several speech recognition tasks. Their superior performance is due to their convolutional structure that processes several, slightly shifted versions of the input window using the same weights, and then pools the resulting neural activations. This pooling operation makes the network less sensitive to translations. The convolutional network results published up till now used sigmoid or rectified linear neurons. However, quite recently a new type of activation function called the maxout activation has been proposed. Its operation is closely related to convolutional networks, as it applies a similar pooling step, but over different neurons evaluated on the same input. Here, we combine the two technologies, and experiment with deep convolutional neural networks built from maxout neurons. Phone recognition tests on the TIMIT database show that switching to maxout units from rectifier units decreases the phone error rate for each network configuration studied, and yields relative error rate reductions of between 2% and 6%.

**Index Terms**: Deep neural networks, convolutional neural networks, maxout networks, TIMIT

## 1. Introduction – Relation to Prior Work

Excellent speech recognition results have lately been reported using deep Convolutional Neural Networks (CNNs) [1, 2, 3, 4, 5, 6]. Compared to standard fully-connected neural networks, the main difference is that CNNs process the input in small localized patches, looking for the presence of relevant local features. These local feature detector neurons are evaluated at several slightly different positions, and the resulting activations are pooled. This pooling step makes the network less sensitive to small translations, and hence the choice of the pooling function plays an important role. In general max-pooling is applied [1], but more sophisticated pooling functions have also been suggested, and some of these have already been tried out in speech recognition as well [5, 3].

In speech recognition the two axes of a spectro-temporal representation have different roles, and should be handled differently. By applying convolution along the frequency axis, we can have acoustic models that are more robust to speaker and speaking style variations. In all the studies that experimented with frequency-domain convolution, researchers found that CNNs consistently outperform fully connected deep neural networks (DNNs) on the same task [1, 2, 4, 6].

This publication was supported by the European Union and co-funded by the European Social Fund. Project title "Telemedicine-oriented research activities in the fields of mathematics, informatics and medical sciences", project number: TÁMOP-4.2.2.A-11/1/KONV-2012-0073.

The advantage of allowing small shifts along the time axis is less obvious, as hidden Markov models inherently handle time shifts. Indeed, the recent studies by Abdel-Hamid et al. and Sainath et al. found that pooling along the time axis provides only negligible benefits [3, 5]. However, it is still advantageous to process a long time-span of input in smaller local chunks. Motivated by this fact, Veselý proposed a special convolutional structure that performs convolution along the time axis, with downsampling playing the role of pooling [7, 8]. We have recently shown that the frequency-domain convolution technique of Abdel-Hamid and Sainath can be nicely combined with the time-domain convolution method of Veselý et al. With the combined method, we reported a record-breaking phone recognition accuracy score on TIMIT [6].

All the studies cited above built the convolutional networks out of sigmoid neurons [1, 4], or rectified linear units [5, 8, 6]. However, now a novel type of neural activation function called the maxout activation has been proposed [9]. Instead of processing the activation of each neuron separately, as is usual, this function takes a group of neurons, and outputs the maximum of their activations. Maxout networks have already been tried out in speech recognition, and achieved good results [10, 11, 12]. Since then, several refinements to the maxout function have also been suggested [13, 14].

The max-pooling procedure of convolutional networks and the maximization applied in maxout networks are very similar. They differ only in their input, as convolutional nets pool the outputs of the *same neuron* evaluated over *different input vectors*, while maxout networks pool the outputs of *different neurons* evaluated on the *same input*. However, the maximization step itself is technically the same. Thus, it is a natural idea to combine the two techniques, and construct convolutional networks out of maxout neurons. Yet, to our knowledge, this possibility has not been explored this far. In this article we build such deep convolutional maxout networks, and evaluate their performance on the TIMIT database. As a basis of comparison, our earlier paper will serve as the baseline, where we used the same network structure, but with rectified linear units [6].

## 2. Convolutional Neural Networks

The operation of the convolutional neurons of a CNN differs from standard neural units in three key ways, which can be summed up by the words 'locality', 'weight sharing' and 'pooling' [1]. Firstly, while conventional neural nets are usually fully connected, CNNs process their input in small local patches. Because of this requirement of locality, they are trained on a time-frequency representation instead of the classic MFCC features. In our case, the input to the network consists of the energy levels of 40 mel filter bank channels. These 40 mel-channels are divided into 7 wider frequency bands that each cover 7 mel-

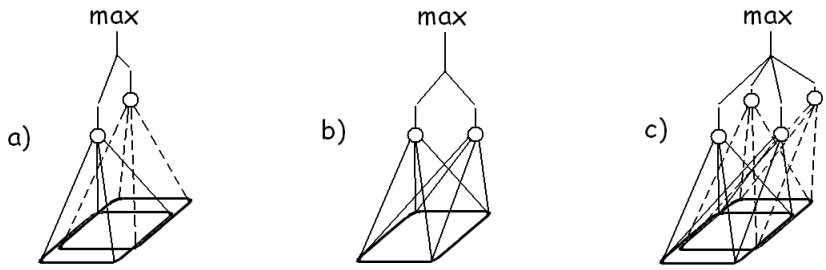


Figure 1: (a) Convolutional neurons pool the results of processing different input blocks using the same weights (weight sharing is indicated by dashed lines). (b) Maxout networks pool the results of processing the same input data using different weight vectors. (c) The proposed convolutional maxout network performs both types of pooling jointly, in one go.

channels. The input is processed in 17-frame blocks along the time axis, so each convolutional neuron operates on a 7x17 spectro-temporal window.

Secondly, the convolutional units get evaluated on several, slightly shifted versions of their input window. These blocks are processed using the same weights, which feature is referred to as 'weight sharing' (see Fig. 1a). In speech recognition experiments the shifting is usually applied only along the frequency axis. In our implementation, the amount of shifting will be measured in mel channels. For example, a pooling size $r$ will mean that the convolutional units process $r$ versions of their input window shifted by $0, 1, .., r-1$ mel banks.

Thirdly, the neural activations got at the various positions are turned into one value in the 'pooling' step. Several strategies exist for this, the classic one being max-pooling [1], but other, more sophisticated pooling formulas have also been proposed. For example, Abdel-Hamid et al. investigated weighted softmax pooling [3], while Sainath et al. studied $p$-norm pooling and stochastic pooling [5]. However, they did not achieve significantly better results with these methods than those with simple max-pooling. Here we will apply max-pooling, because of its similarity with the maxout activation function.

Having discussed the operation of convolutional neurons, let us describe the structure of the whole network. First, each spectral band is processed by a set of convolutional neurons. There are two strategies for combining the information got from the neurons assigned to different spectral regions. Abdel-Hamid et al. argue that the spectral phenomena occurring in different spectral regions are different, and so they apply weight sharing in a limited way, only within the spectral bands [1]. However, Sainath et al. showed that full weight sharing may also give similarly good results [5]. Here we apply limited weight sharing, so our models assign different sets of neurons to the seven spectral regions. Their output is concatenated and processed further by three additional, fully connected layers. We should mention here that it is also possible to stack several convolutional layers on each other. For example, Sainath et al. achieved the best performance with a model of 2 convolutional layers plus 4 fully connected layers [5].

### 2.1. Convolution along the time axis

The early studies on CNN-based speech recognition applied convolution just along the frequency axis [1, 4]. The advantage of extending the convolution to the time axis as well is less obvious, as hidden Markov models inherently handle time shifts. Recently, both Abdel-Hamid et al. and Sainath et al. experimented with convolution along time, and the improvements indeed proved negligible [3, 5].

Independently of these teams, Veselý et al. developed a different network architecture that performs convolution along time [7, 8]. This architecture was motivated by the success of hierarchical ANN models, where a neural network is trained on a block of acoustic feature vectors, and then a second network is trained on a block of posterior output vectors got from the first network [15, 16]. Veselý showed that even better results can be obtained if the two networks are trained as one unit, by propagating the error down from the upper to the lower network. In this hierarchical model the lower sub-network processes only a subset of the input of the whole, joint structure, so the requirement of locality is fulfilled. Several, shifted versions of the local input window get processed using the same sub-network, so weight sharing is also present. Lastly, the output of these local sub-networks is downsampled before being combined by the higher-level layers. Hence, this model can indeed be called convolutional if we regard downsampling as a special kind of pooling function. As we got good results with this sort of architecture earlier [8], we will use it here as well. However, our goal here is to exploit the similarity between max-pooling and maxout units. Since Veselý's model has no actual pooling procedure, in this study we will prefer to interpret this technology as a refined hierarchical model rather than a convolutional model.

## 3. Maxout Neural Networks

The output of a classic perceptron-type neuron is defined as

$$o = \phi(z), \qquad z = \mathbf{w} \cdot \mathbf{x} + b \ .$$

That is, first the linear activation $z$ of the neuron is calculated from the input vector $\mathbf{x}$ , the weight vector $\mathbf{w}$ and the bias term $b$, and then $z$ gets transformed by the nonlinear activation function $\phi$. In conventional networks the sigmoid function is used as $\phi$, but recently novel types of functions such as the rectifier function $max(z, 0)$ have been proposed for this purpose [17]. In speech recognition, rectified linear (ReLU) units were found to be efficient building blocks for deep neural network based acoustic models [18, 19, 20, 21].

Goodfellow et al. suggested generalizing the rectifier function so that the maximum is taken over the linear activation of several neurons [9]. That is, the proposed maxout function divides the $N$ neurons of a given layer into $L$ groups of size $K$ ($N = K \cdot L$), and the output of the $l$th group is calculated as

$$o_l = \max_{k=0}^{K-1} z_{lK+k}, \qquad l = 0, ..., L-1 \ .$$

Several studies have already investigated the applicability of the maxout activation in speech recognition. All these studies found that maxout networks perform better or at least no worse

| Network type and training method | frame error on dev. set | phone error | |
|---|---|---|---|
| | | dev. set | core test |
| ReLU | 36.4% | 18.6% | 20.6% |
| maxout | 35.3% | 17.4% | 20.1% |
| 2-norm, DPT | 37.5% | 17.5% | 20.3% |
| maxout, DPT | 35.0% | 17.1% | 19.4% |
| maxout, mixed DPT | 34.2% | 17.0% | 19.5% |

Table 1: Phone error rates of fully connected ReLU, maxout and 2-norm networks consisting of 4 hidden layers.

than ReLU networks, and the biggest gains were reported under low-resource conditions [10, 11, 12]. However, they also observed that the maxout function is inclined to overfit the data, as the gradient is propagated back only to the neuron that gave the largest activation. One possible remedy is to apply stochastic pooling, which chooses its output (and hence the backpropagation path) randomly, with a probability proportional to the value of the corresponding activation [14]. Another solution is to refine the pooling function itself. Zhang et al. found that $p$-norm pooling gives better results than those for max-pooling [13]. One possible explanation is that $p$-norm acts as a smoothed version of max-pooling, where all the pooled units contribute to the result, and also get updated proportionally to their contribution.

Notice that the pooling performed by the maxout function is technically the same as the pooling step applied in convolutional networks. The difference is that in convolutional networks the pooling is performed over neurons that process *different input* vectors using the *same weights*, while in maxout networks it is applied over *different neurons* that process *the same input* (compare Figs. 1a and 1b). However, the pooling operator does not need to know how the actual values to be pooled were obtained. Hence, it is a natural idea to implement a convolutional layer of maxout units, where the two pooling operations are performed in one go, as illustrated in Fig. 1c. In this study we investigate this technology.

# 4. Experimental Settings

The results we report are phone recognition error rates on the well-known TIMIT database. The training set consisted of the standard 3696 'si' and 'sx' sentences, while testing was performed on the core test set (192 sentences). A random 10% of the training set was held out as the 'development set', which was used for validation purposes, and for tuning the meta-parameters. In all our experiments we used a phone bigram language model estimated from the training data. To get frame-level labels for training, forced alignment was performed with a conventional context-dependent HMM of 858 tied states. These were derived from the 61 phone labels, and they were mapped to the usual set of 39 labels in the evaluation phase. To remain consistent with earlier studies on TIMIT (e.g. [22]), the language model weight and the phone insertion penalty parameters were not tuned, but were just set to 1.0 and 0.0, respectively.

For preprocessing we used the output of 40 mel-scaled filters and the overall energy, along with their $\Delta$ and $\Delta\Delta$ values, altogether yielding $41 \cdot 3 = 123$ features per frame. The same input representation was employed in many previous studies on TIMIT [22, 20, 6, 1]. As input, the fully connected network configurations were trained on 17 consecutive frames of these 123 spectral features. In the case of convolutional networks, the 40 spectral channels were divided into 7 wider frequency bands, which contained 7 mel channels and overlapped by 2 channels

| group size | layer size | dev. error | test error |
|---|---|---|---|
| 2 | 2714 | 17.0% | 19.5% |
| 3 | 3204 | 16.9% | 19.3% |
| 4 | 3584 | 17.0% | 19.3% |
| 5 | 3890 | 17.2% | 19.6% |

Table 2: The effect of the group size on the performance of the maxout network.

(following our earlier study [6]). Each convolutional neuron operated on 17 frames of data coming from one spectral band. For each band, the corresponding 7 mel-channel energy values were extended with the frame-level energy and the derivatives of all these features [1]. This in total gave an input vector of $17 \cdot 8 \cdot 3 = 408$ features per spectral band.

All network configurations were trained using semi-batch backpropagation, the batch size being 100. The training target function to be optimized was the standard frame-level cross-entropy cost. The initial learn rate was set to 0.001 and held fixed while the error on the development set kept decreasing. Afterwards it was halved after each iteration, and the training was halted when the improvement in the error was smaller than 0.1% in two subsequent iterations.

# 5. Results and Discussion

## 5.1. Deep maxout networks

First we sought the best way of training fully connected deep maxout networks, without any convolution being involved. For comparison, a similar ReLU network served as the baseline. This network contained 4 hidden layers with 2000 ReLU neurons per layer [20]. In the first experiment, the ReLU units were replaced by maxout units, using a group size of 2. As maxout units give one output per group and not per neuron, we had to increase the number of neurons per layer to 2714, in order to keep the number of free parameters (weights) the same.

The results we got are listed in Table 1. The first two rows show that maxout networks can indeed significantly outperform ReLU units when using the same simple backpropagation training method. We should add, however, that in the case of the maxout net we had to apply a large momentum (of 0.9) for the good results, while the ReLU network showed no improvement with the use of momentum. The relative error reduction achieved by switching from ReLU to maxout units was 6.5% on the development set and 2.4% on the core test set.

We also performed some experiments with the 2-norm pooling method proposed by Zhang et al. [13]. Unfortunately, we did not get better results with the 2-norm function than with maxout. Our impression was that the normalization layer they apply plays a critical role, and it was not part of our implementation. Also, we achieved reasonable results with the 2-norm function only when applying discriminative pre-training (DPT) [23]. This method builds the network layer by layer, running the backpropagation training for a couple of epochs after the addition of each layer. The third row of Table 1 shows the best result we could obtain with 2-norm networks using DPT. Surprisingly, while the frame error rate attained is much worse than that of maxout, the phone error rates are practically the same.

Next, we wondered whether DPT would also improve the maxout scores. As can be seen in the fourth row, it gave only a slight decrease in the error rate on the development set, though the improvement on the test set is notably larger. Compared to

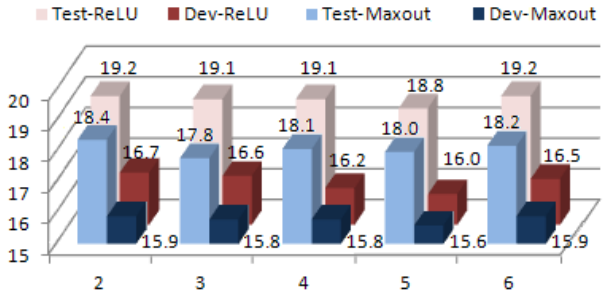

Figure 2: The effect of pooling size on the phone error rate.

the ReLU network, the relative error reduction with this model is 8% on the development set and 5.8% on the test set.

Lastly, we tried to combine the good convergence property of maxout with the smoother behavior of 2-norm. For this purpose, during pre-training, we applied 2-norm pooling instead of max-pooling for random training instances, with a probability of $q$. We hoped that this stochastic addition of the 'flavor' of the smoother 2-norm function to the maxout target function would help the training process avoid local minima. The best results were obtained when setting $q$ to 0.2. As the last row of Table 1 shows, this mixed pre-training method achieved a much better frame error rate than either maxout or 2-norm pretraining. Though the drop of frame error rate is not reflected in the phone-level errors, we opted for using this pre-training method in all the subsequent experiments.

Next, we experimented with varying the group size. We were careful to keep the number of parameters the same, so more neurons were allowed for larger group sizes. As Table 2 shows, we observed no significant changes in the error rate on the development set, so we decided to use a group size of 2. Other researchers also found this group size to be the best [10, 14, 12], with the exception of Zhang, who reported 10 to be the best group size for 2-norm networks [13].

### 5.2. Convolutional deep maxout networks

In the next experiment, the lowest layer of our deep maxout network was replaced by convolutional units. For this purpose, the input was divided into 7 frequency bands, as was explained in Section 4. To keep the number of parameters the same as it was in the fully connected case, 756 convolutional maxout neurons (with group size of 2) were assigned to the processing of each band. The remaining 3 layers of the convolutional network consisted of fully connected maxout neurons. For more details on the convolutional structure see [6].

The experiments sought to answer two questions. First, just as in the fully connected case, we wondered how much gain we could obtain from switching from ReLU units to maxout units. Second, as was explained earlier, our model executes the pooling of the maxout groups and the convolutional outputs jointly, in one step. Thus, we were interested to see how this joint pooling performed when varying the convolutional pooling size $r$.

Fig. 2 shows the phone error rates obtained with various pooling size values. The scores of a ReLU network of the same structure and size are given as reference [6]. As can be seen, the maxout network outperformed the ReLU network for all $r$ values. They both attained the lowest error rate on the development set at $r = 5$, but the scores are quite similar for all $r$ values, for both network types and data sets. Hence, the joint pooling actually worked just as well as the standard convolutional pooling performed in the ReLU network.

| Network type | devel. set | core test set |
|---|---|---|
| Hierarchical, ReLU | 14.2% | 17.6% |
| Hierarchical, maxout | 14.0% | 17.0% |
| Hierarchical, ReLU, dropout | 13.9% | 16.7% |
| Hierarchical, maxout, dropout | 13.3% | 16.5% |

Table 3: Phone error rates obtained with the hierarchical model, without and with dropout.

Compared to its ReLU counterpart, the maxout convolutional network with $r = 5$ attained a 4.3% relative error rate reduction on the development set and 2.5% on the test set. Compared to the best performing fully connected maxout network of Table 1, the convolutional structure brought a relative error reduction of about 8% for both the development and the test sets, again justifying the superiority of convolutional networks over standard fully connected nets.

### 5.3. Constructing a hierarchical model

In the last group of tests, the convolutional maxout network described above was turned into a Veselý-style hierarchical model [7, 8]. For this, a smaller version of the network was constructed, which processes 9 frames of data instead of 17 frames, and its uppermost hidden layer is turned into a 'bottleneck' that contains only 400 units. This network was extended by adding two more hidden layers, which concatenate five output vectors got from the bottleneck layer at five different positions, which are five frames apart. This hierarchical construct lets us expand the time span of the model considerably (in this case to 29 frames) with only a small increase in the number of weights. For more details on this technology, see our earlier article [8].

The results obtained with the hierarchical convolutional model are shown in Table 3. The difference between the ReLU and the maxout results was quite small, only 1.4% relative error rate reduction was achieved on the development set, while an error decrease of 3.4% was obtained on the core test set.

We also tried to train the two networks with the application of the dropout method [24]. Dropout was shown to work nicely with ReLU networks [18, 8, 5], and some researchers have already applied it in the training of maxout networks [12, 14]. We found that the same dropout rate of 0.25 gave the best results for both networks. As shown in Table 3, dropout yielded a somewhat smaller improvement in the case of the maxout network, but this improvement is more balanced between the development and the test sets. To our knowledge, the 16.5% we attained is currently the lowest phone recognition error rate on TIMIT.

## 6. Conclusions and Future Work

Here, we investigated the possibility of building convolutional networks from maxout neurons. What makes this approach interesting is that the pooling operation of convolutional neurons and maxout neurons are closely related. In our solution, the two pooling procedures are executed jointly, in one step. The results of the phone recognition tests on TIMIT revealed that the proposed method works well. In summary, maxout units decreased the error rate for each network configuration examined, and yielded relative error rate reductions of 2% to 6%, compared to ReLU networks with the same size and structure.

In the future, we intend to examine more sophisticated pooling functions such as stochastic pooling [14] and $p$-norm pooling [13]. Also, we plan to extend the tests to larger databases to see how the proposed method works on larger tasks.

# 7. References


[1] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural network concepts to hybrid NN-HMM model for speech recognition," in *Proc. ICASSP*, 2012, pp. 4277 – 4280.

[2] L. Deng, O. Abdel-Hamid, and D. Yu, "A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion," in *Proc. ICASSP*, 2013, pp. 6669 – 6673.

[3] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition," in *Proc. Interspeech*, 2013, pp. 3366 – 3370.

[4] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. ICASSP*, 2013, pp. 8614–8618.

[5] T. N. Sainath, B. Kingsbury, A. Mohamed, and B. et al. Ramabhadran, "Improvements to deep convolutional neural networks for LVCSR," in *Proc. ASRU*, 2013, pp. 315–320.

[6] L. Tóth, "Combining time- and frequency-domain convolution in convolutional neural network-based phone recognition," in *Proc. ICASSP*. 2014, accepted, in print.

[7] K. Veselý, M. Karafiát, and F. Grézl, "Convolutive bottleneck network features for LVCSR," in *Proc. ASRU*, 2011, pp. 42 – 47.

[8] L. Tóth, "Convolutional deep rectifier neural nets for phone recognition," in *Proc. Interspeech*, 2013, pp. 1722–1726.

[9] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. ICML*, 2013, pp. 1319–1327.

[10] M. Cai, Y. Shi, and J. Liu, "Deep maxout neural networks for speech recognition," in *Proc. ASRU*, 2013, pp. 291–296.

[11] Y. Miao, F. Metze, and S. Rawat, "Deep maxout networks for low-resource speech recognition," in *Proc. ASRU*, 2013, pp. 398–403.

[12] P. Swietojanski, J. Li, and J. T. Huang, "Investigation of maxout networks for speech recognition," in *Proc. ICASSP*. 2014, accepted, in print.

[13] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Proc. ICASSP*. 2014, accepted, in print.

[14] M. Cai, Y. Shi, and J. Liu, "Stochastic pooling maxout networks for low-resource speech recognition," in *Proc. ICASSP*. 2014, accepted, in print.

[15] H. Ketabdar and H. Bourlard, "Enhanced phone posteriors for improving speech recognition systems," *IEEE Trans. ASLP*, vol. 18, no. 6, pp. 1094–1106, 2010.

[16] J. Pinto et al., "Analysis of MLP based hierarchical phoneme posterior probability estimator," *IEEE Trans. ASLP*, vol. 19, no. 2, pp. 225–241, 2010.

[17] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. AISTATS*, 2011.

[18] 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, pp. 8609–8613.

[19] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," in *Proc. ICASSP*, 2013, pp. 3517–3521.

[20] L. Tóth, "Phone recognition with deep sparse rectifier neural networks," in *Proc. ICASSP*, 2013, pp. 6985–6989.

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

[22] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. ASLP*, vol. 20, no. 1, pp. 14–22, 2012.

[23] F. Seide, G. Li, L. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011, pp. 24–29.

[24] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.