COMBINING TIME- AND FREQUENCY-DOMAIN CONVOLUTION IN CONVOLUTIONAL NEURAL NETWORK-BASED PHONE RECOGNITION

László Tóth

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

ABSTRACT

Convolutional neural networks have proved very successful in image recognition, thanks to their tolerance to small translations. They have recently been applied to speech recognition as well, using a spectral representation as input. However, in this case the translations along the two axes – time and frequency – should be handled quite differently. So far, most authors have focused on convolution along the frequency axis, which offers invariance to speaker and speaking style variations. Other researchers have developed a different network architecture that applies time-domain convolution in order to process a longer time-span of input in a hierarchical manner. These two approaches have different background motivations, and both offer significant gains over a standard fully connected network. Here we show that the two network architectures can be readily combined, like their advantages. With the combined model we report an error rate of 16.7% on the TIMIT phone recognition task, a new record on this dataset.

Index Terms— Deep neural network, convolutional neural network, rectified linear unit, speech recognition, TIMIT

1. INTRODUCTION – RELATION TO PRIOR WORK

Convolutional Neural Networks (CNNs) have been successfully used in image processing for a long time [1]. However, their applicability to speech recognition had not really been explored before the current renaissance of artificial neural network (ANN) technologies. Compared to standard neural networks, the main difference is that CNNs process the input in small localized parts, looking for the presence of relevant local features. By pooling the output of these local feature detectors, the network can be made more translation tolerant. The CNNs developed for image recognition can be more or less directly applied to a time-frequency representation of a speech signal. However, while in the case of standard images translations along the two axes can be handled similarly, for speech spectrograms shifts along the two axes have quite different meanings. As for the frequency axis, the formant positions of the same phone may vary slightly from speaker to speaker, and also for different speaking styles. Building a model that is less sensitive to these types of variances was the main motivation for applying convolution along *the frequency axis* [2, 3]. All these studies found that CNNs consistently outperform deep neural networks (DNNs) on the same task.

The advantage of allowing small shifts along *the time axis* is less obvious. It can be useful within the range of a couple of frames [4], but beyond that the smearing of the timing information seems to be harmful. Indeed, the recent studies by Abdel-Hamid et al. and Sainath et al. found that convolution along the time axis brings only negligible benefits [5, 6].

Independently of the above teams, Veselý et al. developed a slightly different neural network structure that successfully exploits time-domain convolution [7]. The key to the success of this model is that during pooling the position information is not discarded. Hence in this model the main role of convolution is not shift invariance, but of allowing the model to hierarchically process a fairly long time-span of input. Recently, we implemented a version of Veselý's model using rectified linear neurons, which seem to be better building units of deep networks than the standard sigmoid neurons [8, 9, 10, 11]. Using this model we reported a record phone recognition accuracy on TIMIT [12].

In this paper, we make a natural extension of our earlier work: we combine the frequency-domain convolution technique of Abdel-Hamid and Sainath [2, 3, 5, 6] with the time-domain convolution method of Veselý et al. [7, 12]. To our knowledge, these two approaches have not been combined before. The recognition experiments we report on TIMIT show what can be expected from combining the two techniques, and we also set a new record on this database.

2. CONVOLUTIONAL NEURAL NETWORKS

CNNs differ from standard ANNs in several ways. First, they process their input in small local patches. Because of this requirement of locality, they are trained on a time-frequency

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

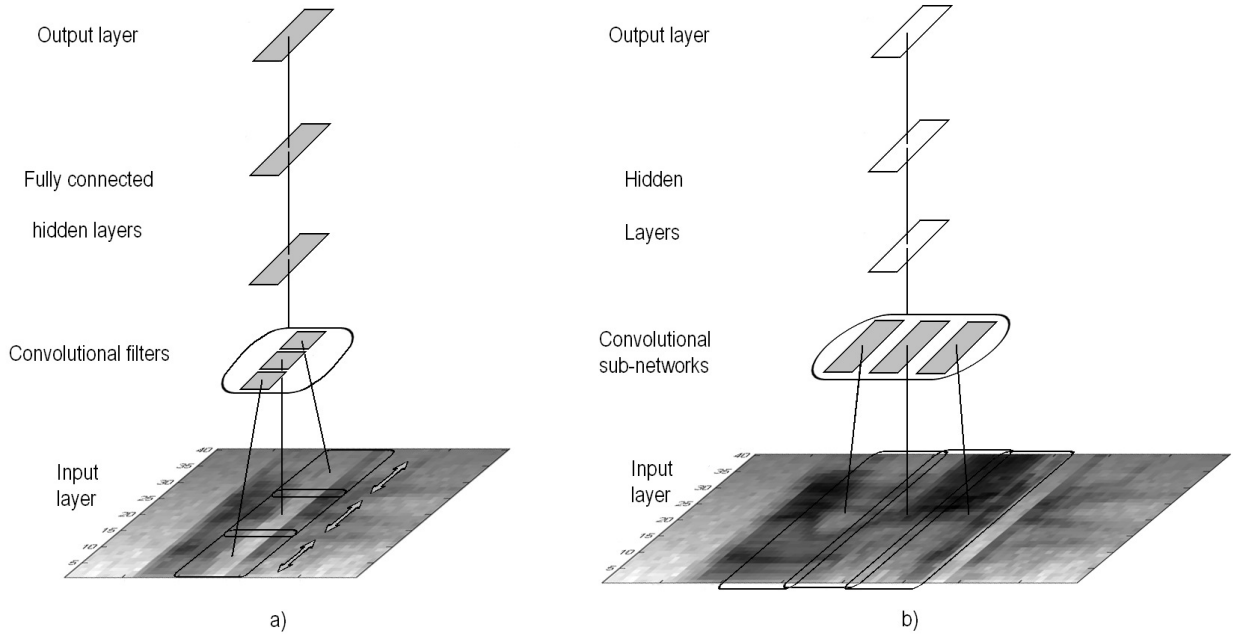

Fig. 1. Illustration of the network structure applied here. For clarity, full connections between layers are denoted here by just a single line. (a) Network topology with frequency-domain convolution only. (b) Network topology with time-domain convolution only. Substituting network (a) for of the 'convolutional sub-networks' (grey boxes) of figure (b) provides the proposed network structure that performs convolution along both axes.

representation instead of the usual MFCC features. Second, the convolutional units process several blocks of input which are slightly shifted in time or/and frequency. These blocks are processed using the same weights, a feature known as 'weight sharing'. Afterwards, the neural activations obtained at the various positions get pooled, for which several strategies exist. These pooled activations may be processed further by additional convolutional layers or by fully connected layers.

2.1. Convolution along the frequency axis

Fig 1a shows the structure of our convolutional network in the case where only frequency-domain convolution is applied. The input to the network consists of the output of 40 mel filter bank channels plus the frame-level energy, along with the corresponding Δ and $\Delta\Delta$ parameters. This input representation is the same as the one we used earlier [8, 12] and it was also applied by other researchers experimenting on TIMIT [15, 2].

The convolutional layer of the network consists of a set of 'filters' that take their input from localized time-frequency blocks. These filters process several, slightly shifted versions of the input window using the same weight set. In the case of frequency-only convolution, shifting is applied only along the frequency axis (symbolized by white arrows in Fig 1a). Then the neural activations obtained from these shifted inputs get pooled. In our implementation, the amount is shifting will be measured in mel channels. For example, a pooling size r will mean that the convolutional layer processes and pools r input windows shifted by $0, 1, .., r$ mel banks. We applied the most popular max-pooling rule [2], though other pooling strategies are also possible [6]. The local input window will be 15 frames along time (like that in [2]), while the optimal size along frequency will be found experimentally.

Along with the size, we will also vary the number of filters used to cover the whole frequency range of 40 mel channels. There are two strategies for combining the information obtained from these different spectral regions. Abdel-Hamid et al. argue that the spectral phenomena occurring at distant frequency regions are different, so weight sharing makes sense only within a limited bandwidth. This reasoning leads to the *limited weight sharing* scheme [2]. However, Sainath et al. showed that *full weight sharing* may also give similarly good results [6]. Here we will apply limited weight sharing, which was found to perform better on TIMIT [2].

Lastly, the output of the convolutional layer may be processed by additional layers. Our model applies three fully connected layers for this purpose, but it is also possible to stack several convolutional layers on each other. For example, Sainath et al. got the best performance with a model of 2 convolutional layers plus 4 fully connected layers [6].

2.2. Convolution along the time axis

The convolutional filters of the model described above could be easily modified so that they perform pooling not only in frequency, but also in time. As the nice results obtained with

time-delay neural networks showed, invariance to small shifts in time may indeed be advantageous [4]. Recently, both Abdel-Hamid et al. and Sainath et al. experimented with convolution along time, but the improvements were much smaller than that got with convolution along frequency [5, 6].

Independently of these teams, Veselý et al. developed a different network architecture that performs convolution along time [7]. This architecture was motivated by the success of hierarchical ANN models, where a neural network is trained on (a context of) acoustic feature vectors, and then a second network is trained on (a context of) posterior output vectors got from the first network [13, 14]. 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. Such a network complex will inherently be deep, as both the upper and lower parts are made of several layers. The performance of such networks can be significantly improved by RBM pre-training [15]. But is was recently found that similarly good results can be obtained without pre-training by building the network out of rectified linear units (ReLUs) [8, 9, 10, 11].

Recently, we implemented a Veselý-type hierarchical network using ReLUs, and we got a recognition accuracy on TIMIT that matched the best previous result [12]. Fig. 1b shows the structure of this network. The input of the network is decomposed into several blocks along the time axis. These get processed by sub-networks that use the same shared weight set at each block position. Then, the output vectors of the sub-networks get concatenated and processed further by the upper part of the network. In our implementation there are 5 local input blocks that each cover 9 frames of input, and they are placed at every 5th frame. The sub-networks consist of 4 fully connected layers (not shown in the figure), while the upper part of the network contains 2 more hidden layers. For more details, see [12].

By comparing the convolution carried out by this model with the operation of the frequency-domain convolutional network, we can see similarities and differences. Using the notation introduced earlier, the time-domain convolutional model applies a pooling size of 1 with full weight sharing among the filters. An obvious difference is that the input blocks are processed by just one layer of neurons in one case, and by a sub-network of several layers in the other.

When comparing our solution with the time-domain convolution scheme applied by others [5, 6], the key difference seems to be that while they put the convolutional filters only at one time position and increase their range by increasing the pooling size r, we place several filters (sub-networks) at different places along time, and this place information is preserved when passing the sub-network's output to the higher layers. In fact, as we use a pooling size of 1, the main role of convolution in our model is not to allow shift invariance, but rather to enable the model to hierarchically process a fairly wide range of input without increasing the number of weights.

Network topology	devel. set	core test set
fully connected	18.6%	20.6%
convolution in time	15.4%	18.7%

Table 1. Phone error rates for the baseline fully connected network and for the net that applies convolution along time.

2.3. Convolution along both time and frequency

Combining the two models described above into one that performs convolution along both axes is straightforward: the network shown in Fig. 1a should be substituted for the sub-networks in Fig. 1b. The merged model will be trained by backpropagation in two phases, as proposed in our earlier study [12]. First, the sub-network is trained, then the output layer is discarded and the full network is constructed with randomly initialized weights in the upper layers. Only the upper part is trained for one iteration, and then the whole network is trained until convergence is reached.

3. EXPERIMENTAL SETTINGS

The results reported 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. To get frame-level labels for training, forced alignment was performed with a conventional context-dependent HMM of 858 tied states. The phone label outputs were mapped to the usual set of 39 labels in the evaluation phase. During decoding a phone bigram language model was used, with the language model weight and the phone insertion penalty parameters set to 1.0 and 0.0, respectively.

The neural networks were trained using semi-batch backpropagation, the batch size being 100. The training target function 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. All the neurons of the networks were rectified linear units, apart from the softmax output layer.

4. RESULTS AND DISCUSSION

A fully connected deep network with 4 hidden layers of 2000 rectified linear units served as our baseline [8]. The first line of Table 1 shows the recognition error rates got by using this network. One may notice that these results are worse than the ones we reported in [8], and the reason is that in that study we used a sparsity penalty function during training, which was not applied here.

Number of filters	Width of filters	Number of units per filter	Error on devel. set
4	12	768	16.6%
5	10	638	16.6%
6	8	554	16.9%
7	7	485	16.6%
8	6	433	16.5%

Table 2. Phone error rates obtained by applying convolution along frequency, using various sets of convolutional filters.

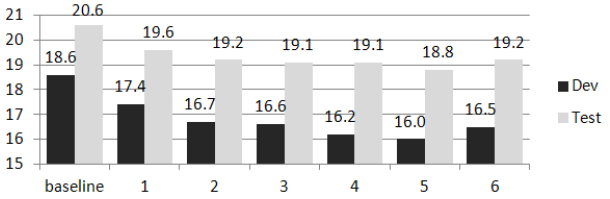

Fig. 2. The influence of pooling size on the phone error rate.

4.1. Convolution along time

The architecture of 1b (with sub-networks of fully connected layers) was investigated in our earlier study [12]. Hence, here we will only repeat the results got with the best configuration. The input to this network consisted of 5 input blocks, each covering 9 frames of input context with an overlap of 4 frames. The sub-network consisted of three layers with 2000 neurons per layer, plus an uppermost 'bottleneck' layer of 400 neurons. The upper part of the network had two further hidden layers of 2000 neurons. The second line of Table 1 shows the results obtained with this network using the two-step training strategy presented briefly in Section 2.3. The error rate improvement over the baseline with this model was approximately 2% absolute (9% relative) on the core test set.

4.2. Convolution along frequency

In the first experiment with this model we attempted to find the optimal number of convolutional filters along the frequency axis. For this purpose, we varied the number of filters between 4 and 8. The size of the filters was chosen so that the neighboring filters overlapped by 2-3 mel channels. The spectral input to each filter was extended by the energy [2]. The filter width was set to 15 frames (the same as that for the baseline system), and the number of neurons in the convolutional layer were always set such that the number of weights remained the same as in the baseline system. The pooling size was always set to 3.

The results obtained on the development set are summarized in Table 2. As can be seen, all the scores are quite similar, so we decided to continue the experiments with the configuration of 7 filters. These filters are 7 mel-channels wide, while Abdel-Hamid uses a filter size of 8 channels [2] and Sainath prefers a filter size of 9 channels [3].

Network topology	devel. set	core test set
convolution along both axes	14.2%	17.6%
the above plus dropout	13.9%	16.7%

Table 3. Phone error rates obtained without and with dropout, using the network that applies convolution along both axes.

The goal of the next experiment was to find the optimal pooling size. As shown in Fig. 2, a pooling size of 5 gave the best result on the development set. In comparison, Abdel-Hamid found $r = 6$ to be optimal for TIMIT [2], while Sainath et al. reported that $r = 3$ performed best on other databases [3]. We think that the optimal value may depend slightly both on the database and the filter size. We note that it also makes sense to combine various pooling sizes within the same model [16]. Similar to convolution along time, the best model reduced the error on the core test set by almost 2% absolute (9% relative), compared to the baseline result.

4.3. Convolution along both time and frequency

Finally, the networks performing frequency-domain convolution and time-domain convolution were combined, and the results are shown in Table 3. Although the score improvements of the two systems did not fully add up, the score of 17.6% we got is more than 1% (5% relative) better than the best scores of the two separate systems.

The training of this merged model was repeated with the application of dropout [18]. Tuning separate dropout parameters for each layer would require sophisticated optimization methods [9], so we simply used the same dropout rate for all layers. The results shown in Table 3 were obtained with a dropout rate of 0.25. On the core test set the error dropped significantly, achieving 16.7%. By comparison, to our knowledge the previous best result was 17.7%, using a special recurrent ANN architecture [17]. The improvement on the development set was much smaller, and a close examination revealed that in this case the number of deletion errors greatly increased. This is caused by a mismatch between the error criteria used during training and testing, which could be avoided by applying a sequence-level training objective [19, 20, 21].

5. CONCLUSIONS AND FUTURE WORK

Here, we proposed a new ANN architecture that combines the time-domain convolution method and the frequency-domain convolution method developed earlier by different authors. We found that the advantages of the two models can be combined, producing a new record of 16.7% on the TIMIT database. However, with the proposed architecture there is still plenty of room for further experimentation. Most importantly, the place and order of information integration along frequency and along time can be varied within the network. We plan to study the effect of such modifications in the future.

6. REFERENCES

[1] Y. Lecun and Y. Bengio, "Convolutional networks for images, speech and time series," in *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib, Ed. 1995, pp. 255–258, MIT Press.

[2] 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.

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

[4] Waibel, A. and Hanazawa, T. and Hinton, G. and Shikano, K and Lang, K. J., "Phoneme recognition using time-delay neural networks," *IEEE Trans. ASSP*, vol. 37, no. 3, pp. 328–339, 1989.

[5] 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.

[6] T. N. Sainath, B. Kingsbury, A. Mohamed, and B. Ramabhadran, "Improvements to deep convolutional neural networks for LVCSR," in *Proc. ASRU*. 2013, 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, "Phone recognition with deep sparse rectifier neural networks," in *Proc. ICASSP*, 2013, pp. 6985–6989.

[9] 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.

[10] 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.

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

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

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

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

[15] 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.

[16] 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.

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

[18] 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.

[19] B. Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proc. ICASSP*, 2009, pp. 3761–3764.

[20] A. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," in *Proc. Interspeech*, 2010, pp. 2846–2849.

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