Acta Cybernetica **22** (2015) 117–134.

Joint Optimization of Spectro-Temporal Features and Deep Neural Nets for Robust Automatic Speech Recognition*

György Kovács† and László Tóth†

Abstract

In speech recognition, feature extraction and acoustical model training are traditionally done in two separate steps. Here, instead, we use a framework that combines spectro-temporal feature extraction and the training of neural network based acoustic models into a single process. We found earlier that this approach can be successfully applied for the recognition of speech. In this paper, we propose two further improvements to our method based on recent advances in neural net technology and extend our evaluation to speech conatminated with new types of noise. By repeating our experiments on TIMIT phone recognition tasks using clean and noise contaminated speech, we can compare the recognition performance of the original framework with our new, modified framework. The results indicate that both these modifications significantly improve the recognition performance of our framework. Moreover, we will show that these modifications allow us to achieve a substantially better performance than what we got earlier.

Keywords: spectro-temporal features, Deep Neural Net, Rectifier Neuron, TIMIT

1 Introduction

One of the biggest challenges for automatic speech recognition (ASR) is to get an acceptable performance even in adverse environments, such as in the presence of background noise. One way of increasing the robustness of ASR systems is to apply spectro-temporal processing methods to the speech signal [9]. In this approach, the features used later in the classification step are obtained by processing small, spectrally and temporally localized patches of the spectrogram. Hence, unlike in

*This publication is 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.

†MTA-SzTE Research Group on Artificial Intelligence, E-mail: {gykovacs,tothl}@inf.u-szeged.hu

DOI: 10.14232/actacyb.22.1.2015.8

traditional processing methods, some of the features may be unaffected by the noise, making the recognizer more robust. The spectro-temporal processing of the patches can be performed by using the two-dimensional discrete cosine transform (2D DCT) or Gabor filters. Good recognition results were reported with both of these approaches earlier [3, 9, 10].

After the initial step of feature extraction, the features are passed on to a machine learning algorithm – in most cases a Hidden Markov Model (HMM) or an Artificial Neural Net (ANN) based recognizer. Traditionally, these two steps (feature extraction and recognition) are performed separately. In earlier papers [11, 12], we showed that the spectro-temporal feature extraction step and the ANN-based recognition step could be combined, and the parameters needed for the two phases could be trained together. Our solution was based on the observation that the spectro-temporal filters can be treated as special types of neurons, and so the standard backpropagation training algorithm of ANNs can be extended to the feature extraction step as well. We experimented with neurons that simulate three types os spectro-temporal feature extraction methods, namely a set of Gabor filters, 2D DCT, and randomly generated filters. Our results told us that in each case, our new method enhanced the performance of the filter sets by extending the scope of the backpropagation algorithm to the neurons simulating them.

In this study, we further improve the performance of our system by incorporating recent advances in the area of neural networks. In standard ANN implementations there are three layers, namely an input layer, an output layer (applying the soft-max nonlinearity), and in between a hidden layer that uses a sigmoid activation function. Recently, it has been shown that a significantly better performance can be achieved by increasing the number of hidden layers [6]. Unfortunately, training these 'Deep' Neural Nets (DNNs) with three or more hidden layers using the classic backpropagation algorithm has certain difficulties. A solution to these problems was given by Hinton et al., leading to a renaissance of ANN-based technologies in speech processing [6]. An even simpler solution was later given with the introduction of rectifier neural networks [14]. Here, we combine this type of DNN with our framework introduced in an earlier paper [11]. We also modified the way convolution is handled by our model. By evaluating our system in phone recognition tasks using the TIMIT speech database, we will show that these techniques improves the performance of our model in the case of clean speech and noise contaminated speech.

Below, we will describe the methods and tools we used in our experiments, then briefly introduce our spectro-temporal filtering methods, and the original framework for the joint optimization of filters and classifiers. This will be followed by a description of the proposed modifications and demonstration of the effect of these on the recognition performance. Lastly, we will draw some conclusions and suggest further ideas for future study.

2 Experimental setup

Before we describe our experiments and present our results, we will first introduce the settings and instruments we applied in our study.

2.1 Pre-processing

First, we computed the spectrograms from sound files with 400 samples (25 ms) per frame, using 160 sample (10 ms) hop size, applying a 1024-point Fast Fourier Transform (FFT) on the frames. The resulting spectrograms were transformed to a logarithmic mel-scale of 26 spectral channels, and they were normalized so as to give a zero mean and unit variance. Afterwards, to avoid the artificial down-weighting of the lowest frequency bins, the four lowest channels of the spectrogram were mirrored.

2.2 HTK Toolkit

After the sound files were pre-processed, our neural net framework was trained to supply posterior probability values for each frame coming from one of the 39 phonetic classes. To create a phone recognizer from this data, we used Hidden Markov Models (HMM) implemented in the HTK toolkit [17], which was modified so as to be able to work with the neural net posteriors. In the literature this construct is known as a HMM/ANN hybrid model [2]. To get our phone recognition results, we also applied a simple bigram language model and used the train set of the speech database to tune our phone insertion penalties.

2.3 Speech database and noises

The database we worked with was the TIMIT speech corpus. In our experiments we followed the standard partitioning of having a train set of 3696 sentences, and a test set of 192 sentences. We used 90% of the train set to train the weights of our neural nets, while the remaining 10% was used as an evaluation set and the stopping criteria. The training of neural nets was performed on clean speech. Testing was performed on both clean and noise contaminated speech.

We employed the FaNT [7] tool to add different types of noises with different signal to noise ratios to the clean speech of the test set. These noise samples mainly came from the NOISEX-92 database [15]. Some of them were artificial, and some came from real-life applications. The two types of artificial noises used were band limited noise, which we created by filtering white noise with a bandpass filter active between 3000 Hz and 5000 Hz, and pink noise coming from the NOISEX-92 database, which has the highest energy at 0Hz and tails off at higher frequencies. Among the noise types coming from real-life applications there was babble noise (simulating the effect of people talking in the background), Volvo noise (recorded in a running car), and also factory noise (which simulates the effect of a nearby production line).

3 Spectro-temporal filters

In spectro-temporal analysis we extract spectro-temporally localized patches from the spectrogram of the speech signal, and create features for ASR purposes by processing them using standard filtering methods. Mathematically, a spectro-temporal feature can be described by the formula

$$o = \sum_{f=0}^{N} \sum_{t=0}^{M} P(f,t) F(f,t), \tag{1}$$

where N and M are the height and width of patch P and filter F, respectively, and they both have to be the same size. There are many different methods available for getting the proper coefficients for the filter $F(f,t)$. Below, we describe two well-known methods, then in Section 4 we present a new method for doing so.

3.1 2D DCT

A common approach is to process the patches using a 2D DCT, which works with the following filter coefficients:

$$F_{pq}(f,t) = \cos \frac{\pi \cdot (f+0.5) \cdot p}{N} \cos \frac{\pi \cdot (t+0.5) \cdot q}{M}, \quad \begin{array}{l} 0 \le q \le N-1 \\ 0 \le p \le M-1 \end{array} \tag{2}$$

where N and M are the respective height and width of the filters for f and t, while p and q specify the modulation frequencies of the filter along the frequency and time axis. Using all possible values of p and q would result in as many features as the number of inputs. However, it is common practice [3] to retain just the output of the filters corresponding to the lowest-order coefficients. And while it is not necessarily the optimal choice, by keeping only 9 coefficients we achieved a performance competitive with the widely used MFCC features [10].

3.2 Gabor filters

Another family of filters that has been applied for feature extraction in speech recognition is Gabor filters [8]. These filters are defined [4] as a product of a two-dimensional Gaussian

$$W(f,t) = \frac{1}{2\pi\sigma_f\sigma_t} e^{-\frac{1}{2}\left(\frac{(f-f_0)^2}{\sigma_f^2} + \frac{(t-t_0)^2}{\sigma_t^2}\right)}, \tag{3}$$

and an oriented sinusoid

$$S_{p,q}(f,t) = e^{j\left(\frac{\pi \cdot f \cdot p}{N} + \frac{\pi \cdot t \cdot q}{M}\right)}, \tag{4}$$

where we iterate f and t over the frequency and time intervals of the patch, and σ_f and σ_t specify the respective bandwiths of the filters. Again, N and M specify

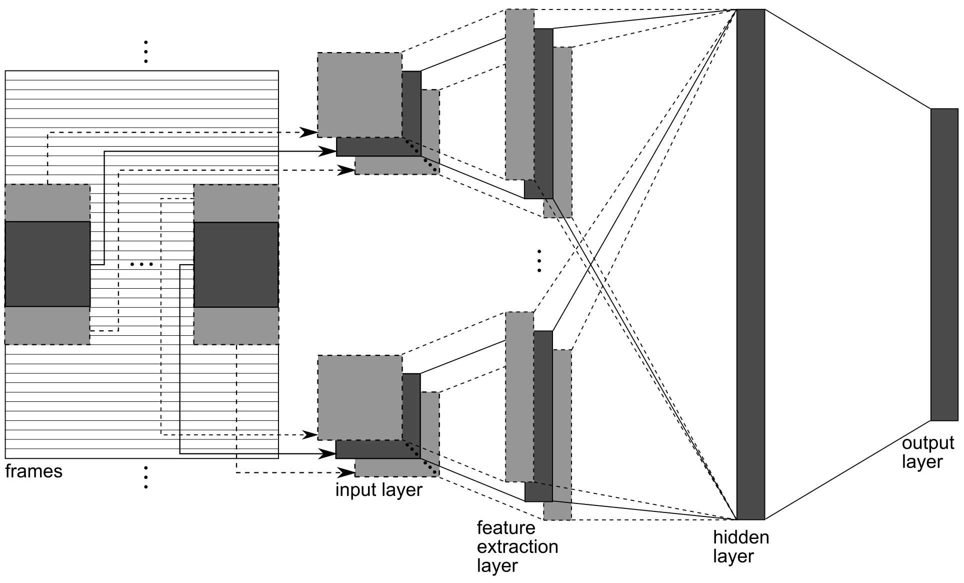

Figure 1: Structure of the ANN for joint feature extraction and classification. The boxes in light grey correspond to additional input blocks or neural units used by the convolutional version of the network.

the transform size, while p and q specify the slanting of the sinusoid as well as its periodicity. These parameters allow many different filters, and unlike in the case of 2D DCT (where there is an assumption about which filters should be kept), the choice of the right Gabor filters for ASR remains an open question [4, 13].

4 Joint optimization of neural net classifiers and spectro-temporal filters

The spectro-temporal features extracted by the filters form the input of a machine learning algorithm. Traditionally, classification or modelling is executed in a separate step before feature extraction. Our proposal was to combine these steps by treating the feature extraction filters as the lowest layer of a neural net, and letting the training algorithm tune the filter coefficients as well. To explain how it was done, let us examine the operation of a simple perceptron model. In general, its output can be obtained using the formula:

$$o = a\left(\sum_{i=1}^{L} x_i \cdot w_i + b\right), \tag{5}$$

where $\mathbf{x}$ is the input of the neuron, L is the length of the input, $\mathbf{w}$ is the weight vector, and b is a bias corresponding to that neuron. Note that if $L = N \cdot M$, and we set a to be the identity function and $b = 0$, and we represent filter F and patch P in (1) in vector form, it becomes apparent that (1) is just a special case of (5). This means that the spectro-temporal filters can be integrated into an ANN classifier system as special neurons, with the filter coefficients corresponding to the weights of the given neuron.

4.1 Structure of the ANN for combined feature extraction and posterior estimation

Fig. 1 shows the proposed structure of the ANN that can perform spectro-temporal feature extraction and classification (phone posterior estimation) in one step. The neural net consists of 4 layers. The first one is the input layer, which extracts 9 by 9 patches (patches with a width of 9 frames, and height of 9 mel channels) from six frequency bands of the spectrogram (with an overlap of 5 channels), and forwards them to the feature extraction layer. In Fig. 1, this is represented by the dark grey areas. The feature extraction layer then takes these patches, and extracts 9 features from each of them (meaning that this layer consists of altogether 54 linear neurons - 6 times 9), and sends the output to the hidden layer (which in our experiments had 4000 neurons [12], using the sigmoid activation function). From this point on, the system behaves just like any other conventional neural net; the hidden layer processes the information and passes it on the the output layer (which in our experiments had 39 neurons, corresponding to the 39 phone classes) that provides the output. Hence, if the weights of the feature extraction layer were initialized with 2D DCT or Gabor filter coefficients, and only the weights of the hidden and output layers were tuned during training, the model would be equivalent to a more traditional system. This means that the structure in Fig. 1 allows our algorithm to evaluate the spectro-temporal features and the ANN in one step, but more importantly, it also allows us to fine-tune the initial coefficients, and so fine-tune the above-mentioned features. If we initialize the coefficients randomly (as is usual in neural networks), it also allows us to fine-tune or evaluate a set of random filters. We showed earlier that the joint optimization of this model outperforms the standard two-step training procedure [11].

4.2 Convolutional neural nets

It is well known that integrating a longer temporal context into the acoustic features can significantly improve the recognition performance. In HMM-based recognition the Δ and $\Delta\Delta$ features are used for this purpose, while in HMM/ANN hybrids it is common to use several neighbouring acoustic vectors [2]. Although spectro-temporal features process longer time intervals than traditional techniques (such as MFCC), we observed that adding the delta features to the feature set improved the results [10]. Although incorporating the delta features into the joint model would be technically challenging, training the network on several neighbouring feature vectors instead of just one is possible by modifying the proposed structure and creating a convolutional neural net [1, 16]. This modification is shown in Fig. 1

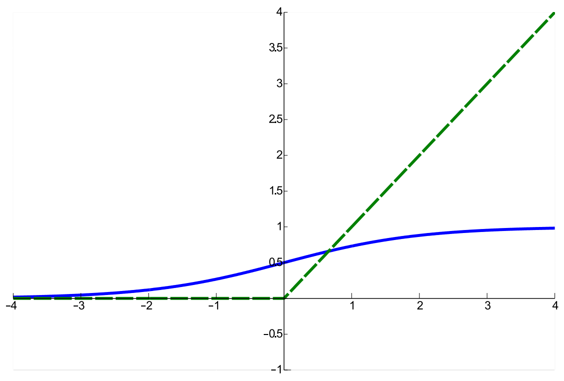

Figure 2: Plot of the sigmoid activation function (solid line), and the rectifier activation function (dashed line).

by the boxes drawn in light grey. As can be seen, in the convolutional networks the feature extraction layer operates on several input patches instead of just one. We should add that the same weights were applied on each input block, so the number of weights did not change in this layer. Evidently, the number of feature vectors processed by the hidden layer increases, but in other respects the hidden and output layers work just as before. Note also that although the patches used do not necessarily have to be immediate neighbours, for simplicity we used only neighbouring patches in an earlier study [11]. Here, we will examine the effect of decreasing the overlap of the processed patches.

5 Deep Rectifier Neural Nets

In contrast to our previous study, one of the two modifications we propose here is to turn the ANN of our joint optimisation model into a deep rectifier neural net. This modification entails altering the activation function, and the structure of the neural net as well. Below, we will elaborate on the modifications made.

5.1 Rectifier neural net

Rectifier neural nets differ from their conventional counterparts only in the activation function used by the neurons in the hidden layer(s). While standard methods use the sigmoid activation function, rectifier nets apply the rectifier function. The difference between these functions can be seen in Fig. 2. The rectifier function ($rectifier(x) = max(0, x)$) has two interesting properties. The first one is its linearity for positive input. Because of this, neurons using this activation function do not saturate as their activity increases. This means that even with multiple hidden layers, the problem of vanishing gradients can be reduced or avoided altogether [14].

Another interesting property is that for all negative values the function returns a constant value of zero. This means that the resulting neural net is usually more sparse, which has computational advantages. Also, sparsitiy is biologically more plausible [5].

5.2 Structure of the DNN for combined feature extraction and posterior estimation

With the replacement of sigmoid neurons with rectifying ones, we can if we wish have more hidden layers in our neural net without having to apply a pre-training algorithm. Here in our experiments, we chose to replace the hidden layer with 4000 sigmoid neurons by three hidden layers, each one consisting of a 1000 rectifying neurons. There were two main reasons for doing this. One was that with this choice, we managed to keep the parameter number of the new framework ($\sim$2.5 million) close to the parameter number of the original one ($\sim$2.1 million). The other was that in our preliminary experiments the progression from two rectifier layers with 1000 neurons to three rectifier layers with the same size was the last modification to provide a significant improvement in the frame level error rates of the validation set. When we went from three rectifier layers with 1000 neurons to four rectifier layers with the same size, this sometimes harmed the error rates, and even when it led to an improvement, the difference was not always significant.

5.3 Results

To evaluate the effect of the new neural net structure on the phone recognition accuracy scores, we carried out several experiments on the TIMIT speech database. First, we will describe the phone recognition results got on clean speech, then present the results got on speech contaminated with artificial noises, and after we will present the results got for speech contaminated with noise samples coming from real-life applications. As in a previous paper using this framework [11], we again used two settings: first the filter coefficients were left unaltered, and in the second setting we trained them. This way, we could learn whether the change in the training method influenced the performance of the two settings. It should be mentioned that all our results came from taking the average of ten independent neural net training results, and we decided that a difference between two averages was significant when the p-value resulting from the two-tailed student's t-test with unequal variance was below 0.05.

5.3.1 Experiments on Clean Speech

As can be seen in Table 1, for the DNN framework, just like in the original framework, the results we got by training the filter coefficients are better than the results we got when the filter coefficients were unaltered. We also see that regardless of the number of layers acted on by the backpropagation algorithm, the DNN framework

Table 1: Phone recognition accuracy scores got on the clean test set of TIMIT.

Initial	Original framework		DNN framework	
Filter weights	unaltered	trained	unaltered	trained
Gabor	73.49%	74.31%	75.58%	76.63%
2D DCT	72.69%	73.88%	75.00%	75.85%
Random	72.99%	73.72%	74.65%	76.42%

always gives better recognition scores than those for the corresponding shallow version. The average relative error rate reduction in the unaltered settings is 7.50%, while in the trained version it is 8.96%.

5.3.2 Experiments on Speech Contaminated with Artificial Noise

We repeated the experiments of clean speech on noise contaminated versions of the TIMIT test set. Here, the models we evaluated were trained on the clean version of the train set, and only the test set was contaminated with noise. First, we used two artificial noise types for this, namely Band-limited noise and pink noise. The numerical results of these experiments are listed in Table 2. Because we have much more data here, a detailed analysis becomes more complex as well. So let us examine the questions raised in the last section separately.

The first question is about how the recognition performance varies in the case where the filter coefficients are unaltered compared to the case where the filter coefficients were also trained. To make the comparison easier, in each unaltered/trained pair of Table 2, when there was a significant difference, the better recognition score

Table 2: Phone recognition accuracy scores got on the test set of TIMIT contaminated with Band-limited and Pink noise with different Signal to Noise (S/N) ratios.

Noise	S/N	Initial	Original framework		DNN framework	
type	ratio	Filter weights	unaltered	trained	unaltered	trained
Band-limited	10db	Gabor	45.76%	**47.47%**	46.36%	**47.21%**
		2D DCT	47.74%	47.57%	**48.52%**	46.52%
		Random	44.46%	46.35%	45.99%	46.14%
	20db	Gabor	60.58%	60.13%*	58.03%	58.11%
		2D DCT	60.01%	**60.71%***	**58.92%**	57.89%
		Random	54.89%	**59.32%**	58.58%	57.80%
Pink	10db	Gabor	32.16%	**35.97%**	37.34%	**38.39%***
		2D DCT	32.23%	**35.24%**	35.73%	**37.19%***
		Random	29.29%	**34.60%**	35.70%	**37.04%***
	20db	Gabor	53.40%	**56.87%**	59.32%	**60.05%***
		2D DCT	51.84%	**55.70%**	57.99%	**59.28%***
		Random	47.75%	**54.98%**	56.73%	**59.66%***

is emphasized in bold. As can be seen, the two types of artificial noise behave quite differently. With pink noise we see the same thing that we saw in the Clean settings case, namely that the trained version always yielded significantly better recognition rates. With band-limited noise, however, in most cases there is no significant difference between the trained and unaltered version. And when there is, usually the trained version is better for just the original framework.

The second question is about how the recognitition performance changes when moving from the original framework to the DNN framework. In this case, to make the comparison easier, in each row of Table 2 where there is a significant difference between the best result got with the original framework, and the best result got with a DNN framework, the framework with a better performance is denoted by an asterisk. As before, we also observe a big difference between the two artificial noise types. In the case of Pink noise, the recognition scores got by the DNN framework are significantly better than those got by the original framework; but this is not so in the case of Band-limited noise. With the latter type there is usually no significant difference in the performance, and in both cases where the difference is significant, the original framework tends to produce better results.

From our findings (see Table 2), we can say that with Pink noise the results are what we expected them to be: the DNN framework not only outperforms the original one, but it also consistently gives significantly better scores with trained filter coefficients. This is not the case with Band-limited noise, where the DNN framework does not provide significantly better results than those got with the original one, and it also tends to produce better scores with unaltered filter coefficients than with trained ones. Lastly, we should mention that with Band-limited noise we got the best result with 2D DCT, while with Pink noise it was Gabor filters that provided the best recognition score.

5.3.3 Experiments on Speech Contaminated with Real-Life Noise

We also conducted experiments with noise types got from real-life applications, the results of which are summarized in Table 3. As we have too much data to overview at once, let us once again examine each of the above questions in turn.

As in Section 5.3.2, we first compare the recognition scores got when the filter coefficients were unaltered compared to the case where the filter coefficients were also trained. And as we did previously, in each unaltered/trained pair of Table 3, where there was a significant difference, the better recognition score is shown in bold, for easier visualization. We see in Table 3 that for Factory and Volvo noise, the DNN framework with trained filter coefficients provides better recognition rates than with filter coefficients left unaltered. The picture is a slightly less clear with Babble noise, but even with this noise type we can say that usually with this framework we got better recognition scores by the training of filter coefficients than without it. When we compare the two frameworks, we see a clear tendency. In each of the eighteen cases, without exception, the best result was got with the DNN framework.

To summarize our findings presented in tables 1,2 and 3, we can say that with

Table 3: Phone recognition accuracy scores got on the test set of TIMIT contaminated with Babble, Factory and Volvo noise, with different Signal to Noise (S/N) ratios.

Noise type	S/N ratio	Initial Filter weights	Original framework unaltered	Original framework trained	DNN framework unaltered	DNN framework trained
Babble	10db	Gabor	45.64%	45.84%	49.39%	49.27%*
		2D DCT	44.10%	**44.96%**	**48.67%**	47.25%*
		Random	43.26%	43.75%	**49.56%**	46.62%*
	20db	Gabor	62.58%	**63.05%**	66.59%	**66.91%***
		2D DCT	61.12%	**62.39%**	65.41%	**65.83%***
		Random	61.70%	**62.36%**	65.45%	**66.58%***
Factory	10db	Gabor	37.63%	**41.94%**	43.93%	**45.27%***
		2D DCT	38.27%	**41.32%**	43.34%	**44.42%***
		Random	41.15%	41.05%	43.70%	**45.40%***
	20db	Gabor	58.11%	**61.14%**	64.07%	**64.66%***
		2D DCT	57.84%	**60.53%**	62.83%	**63.68%***
		Random	59.89%	60.06%	62.57%	**64.48%***
Volvo	10db	Gabor	66.95%	**67.39%**	71.11%	**71.74%***
		2D DCT	67.03%	67.21%	70.47%	**70.81%***
		Random	**67.69%**	66.14%	70.63%	**71.41%***
	20db	Gabor	70.41%	**71.52%**	73.68%	**74.47%***
		2D DCT	70.29%	**71.30%**	73.33%	**74.00%***
		Random	70.93%	70.78%	73.15%	**74.46%***

the exception of Band-limited noise, the DNN framework not only provides better results than the original framework, but in most cases it also yields better recognition scores when the filter coefficients are trained. We also see that usually the best performance was achieved when we employed Gabor filters.

6 Fine-Tuning Convolution

In both the original framework and the DNN framework, we utilized the simplest version of convolution where the patches used were direct neighbours of each other. Furthermore we combined the convoluted layers immediately after the feature extraction layer. This, however, is not necessarily the optimal solution. Hence, below we will describe experiments where we sought to fine-tune the convolutional parameters and the structure of our model.

6.1 Adjusting the Convolutional Structure of the Model

So far, we have been combining the results got from the convolutional feature extraction layer in the first rectifier layer. It is possible though, and it may be beneficial, to perform convolution in one of the rectifier layers as well, and combine the

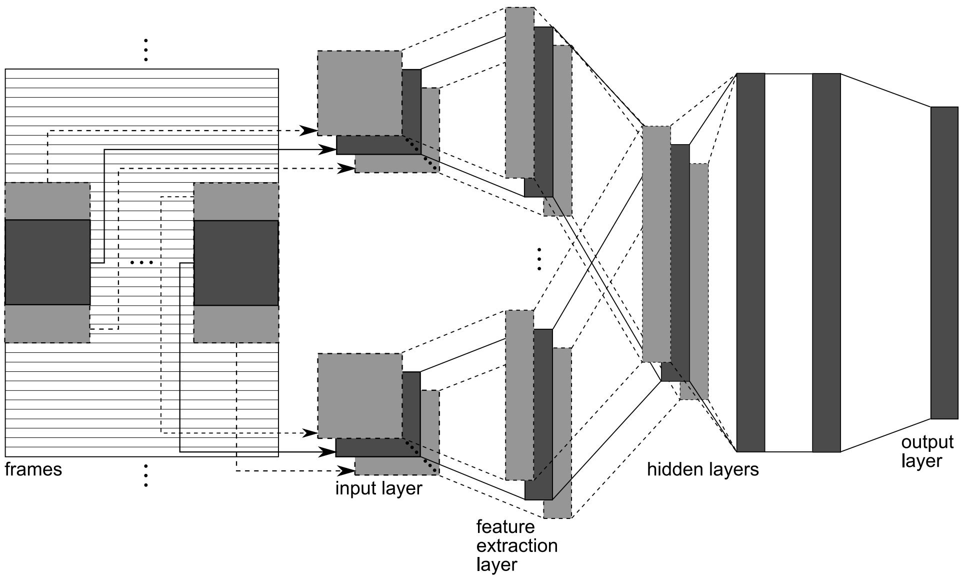

Figure 3: Structure of the deep neural net with modified convolution (DNN-Conv) for joint feature extraction and classification. Here, the new elements in the figure are the three hidden (rectifier) layers, one of which applies convolution.

outputs later. This, however, would lead to a massive increase in the parameters of the model, since the number of outputs in the hidden layer would then be the same as the number of neurons in that layer times the number of neighbouring patches used, which in our case is 9000 (1000 times 9). This (in combination with the 1000 neurons in the next layer) would increase the number of parameters by ~7.5 million (an increase of approximately 200%). To avoid this, here we propose two further modifications. First, instead of using all four neighbours on both sides, we will use only two in such a way that we skip every second patch, which means that the neural net takes its input from the same time-span as before. This modification will reduce the number of extra parameters to ~3.5 million (an increase of approximately 40%). If we also reduce the size of the convolutional rectifier layer from 1000 neurons to 200, we can further reduce this number, eventually making the parameter count of the DNN framework quite similar to the parameter count of the original framework (~2.1 million). The structure of the new neural net framework with the proposed modifications (DNN-Conv framework) can be seen in Fig. 3.

6.2 Number of Neighbours Used

Earlier, we limited the number of neighbouring patches used in convolution by just applying every second patch. As our preliminary results seemed quite promising,

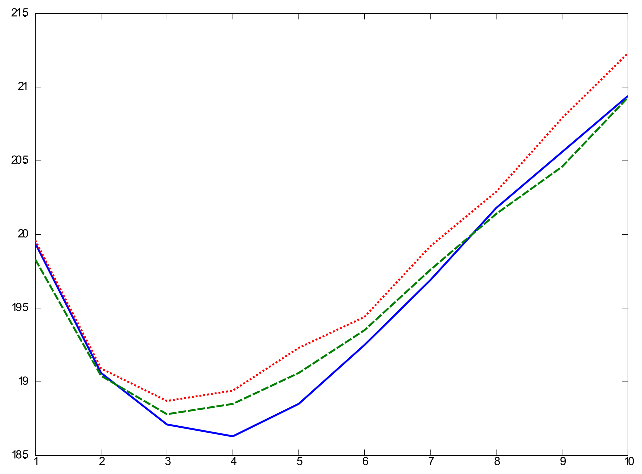

Figure 4: The frame level error rates of the validation set as a function of skipped patches in neural nets using Gabor (solid line), 2D DCT (dashed line) and Random (dotted line) filters.

this raised the question of how many patches should be skipped for an optimal performance. To answer this question, we decided to independently train ten neural nets, each with patch omissions from 1 to 10, using Gabor, 2D DCT and Random spectro-temporal filters (300 neural nets altogether). The average frame level error rates of these experiments are shown in Fig. 4. As can be seen, the best results were achieved with 2D DCT and Random filters (with a 18.78% and 18.87% error rate, respectively) when skipping three patches between the patches used. With Gabor filters some additional improvement could be achieved by skipping 4 patches instead of three (18.63% instead of 18.71%), but as this difference was not significant, we chose to leave out 3 patches for each model version investigated in all our subsequent experiments.

6.3 Results

To evaluate the effect of the proposed changes on the phone recognition accuracy scores of the framework, we repeated our experiments on the TIMIT speech database. As in Section 5.3, first we will describe the phone recognition results got on clean speech, then we will discuss the results got on speech contaminated with artificial noise; and after we will present the results got on speech signal contaminated with noise types arising in real-life applications. Because in the previous experiments the configuration where the filter coefficients were also trained by the

Table 4: Phone recognition accuracy scores got on the clean test set of TIMIT.

Initial Filter weights	Original framework	DNN framework	DNN-Conv framework
Gabor	73.76%	76.63%	**77.02%**
2D DCT	73.46%	75.85%	**76.78**%
Random	73.47%	76.42%	**76.75**%

backpropagation algorithm yielded a significantly better performance in most cases, here we only experimented with this setting. We will compare the results got with the best results got with the DNN and the original framework (regardless of whether the result was got using unaltered filter coefficients or trained ones).

6.3.1 Experiments on Clean Speech

The results we got from applying our procedure on the uncontaminated version of TIMIT test set are summarized in Table 4. As can be seen, the proposed modifications in the framework produced phone recognition accuracy scores that are not just better than those got using the original framework, but also better than those got using to the DNN framework. Also, the improvement in the recognition scores is significant in each case.

Table 5: Phone recognition accuracy scores got on the test set of TIMIT contaminated with Band-limited and Pink noise, with different Signal to Noise (S/N) ratios.

Noise type	S/N ratio	Initial Filter weights	Original framework	DNN framework	DNN-Conv framework
Band-limited	10db	Gabor	47.47%	47.21%	**50.25%**
		2D DCT	47.74%	48.52%	**50.54%**
		Random	46.35%	46.14%	**49.68%**
	20db	Gabor	60.58%	58.11%	**61.90%**
		2D DCT	60.71%	58.92%	**62.52%**
		Random	59.32%	58.58%	**62.30%**
Pink	10db	Gabor	35.97%	**38.39%**	**38.55%**
		2D DCT	35.24%	37.19%	**38.67%**
		Random	34.60%	37.04%	**38.13%**
	20db	Gabor	56.87%	**60.05%**	**60.11%**
		2D DCT	55.70%	59.28%	**59.91%**
		Random	54.98%	**59.66%**	**59.41%**

Table 6: Phone recognition accuracy scores got on the test set of TIMIT contaminated with Babble, Factory and Volvo noise, with different Signal to Noise (S/N) ratios.

Noise type	S/N ratio	Initial Filter weights	Original framework	DNN framework	DNN-Conv framework
Babble	10db	Gabor	45.87%	49.39%	**50.30%**
		2D DCT	44.96%	48.67%	**49.40%**
		Random	43.75%	**49.56%**	48.54%
	20db	Gabor	63.05%	66.91%	**68.20%**
		2D DCT	62.39%	65.83%	**67.83%**
		Random	62.36%	66.58%	**67.38%**
Factory	10db	Gabor	41.94%	**45.27%**	**45.09%**
		2D DCT	41.32%	44.42%	**45.11%**
		Random	41.15%	**45.40%**	44.87%
	20db	Gabor	61.14%	64.66%	**65.18%**
		2D DCT	60.53%	63.68%	**64.68%**
		Random	60.06%	**64.48%**	**64.42%**
Volvo	10db	Gabor	67.39%	71.74%	**72.48%**
		2D DCT	67.21%	70.81%	**72.14%**
		Random	67.69%	71.41%	**72.36%**
	20db	Gabor	71.52%	74.47%	**75.25%**
		2D DCT	71.30%	74.00%	**75.08%**
		Random	70.93%	74.46%	**75.11%**

6.3.2 Experiments on Speech Contaminated with Artificial Noise

Here, we used the version of the TIMIT test set that was contaminated with artificial noise types. The results of these experiments are summarized in Table 5. For better visualization, in each row we highlighted the best phone recognition accuracy score and the ones that do not significantly differ from it in bold. As can be seen, with Band-limited noise the framework with our two modifications significantly outperforms every other framework tested. Pink noise is not much different from this, as in that environment the recognition scores we got with the new framework were always significantly better than those got with the original framework. When these results are compared with those got using the DNN framework, usually they were better as well, with the exception of three cases where there was no significant difference.

6.3.3 Experiments on Speech Contaminated with Real-Life Noise

Lastly, we repeated our experiments on speech contaminated with noise types arising from real-life applications. The results given in Table 6 tell us that with Volvo noise the results got with the DNN-Conv framework surpass even the best results got with the Original framework or the DNN framework. The picture is not so clear with Babble and Factory noise, but usually with those settings the best results were got by applying the DNN-Conv framework. We also see that when there was substantial difference between the best results that were obtained with the DNN-Conv framework using different filter sets, in most cases the best scores were got from using the Gabor filter set.

7 Conclusions and Future Work

Here, we presented our framework for the joint optimization of neural nets and spectro-temporal filters. This framework was described in an earlier paper and here we proposed two refinements to the framework to improve its performance. While the first modification brought about an improvement in the recognition scores compared to the original framework, we got the best performance when we applied both refinements. As we saw, the scores provided by the framework that applied both modifications were significantly better than those got using our original framework, regardless of the filter set applied or the noise used to contaminate the speech signal.

In the future we would like to improve our neural net framework still further. First, we could improve the ASR performance of the system by simulating the Δ and $\Delta\Delta$ features with neurons, as we did when we simulated the filtering with neurons. Second, we would like to find out whether it is possible to efficiently combine this framework with multi-band speech recognition. The features obtained from different bands could then be fed to separate classifiers, and after the resulting posterior values could be combined and become the input for another neural net.

References

[1] Abdel-Hamid, O., Mohamed, A-R., Jiang, H., and Penn, G. Valamivalamivalamiapplying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4277–4280. IEEE, 2012.

[2] Bourlard, H. A. and Morgan, N. *Connectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[3] Bouvrie, J., Ezzat, T., and Poggio, T. Localized spectro-temporal cepstral analysis of speech. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4733–4736. IEEE, 2008.

[4] Ezzat, T., Bouvrie, J., and Poggio, T. Spectro-temporal analysis of speech using 2-D Gabor filters. In *Proceedings of Interspeech*, pages 506–509. International Speech Communication Association (ISCA), 2007.

[5] Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In Gordon, Geoffrey J. and Dunson, David B., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[6] Hinton, G., Deng, L., Yu, D., Abdel-rahman, M., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Dahl, G., and Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[7] Hirsch, H.-Guenter. *FaNT - Filtering and Noise Adding Tool.* Hochschule Niederrhein University of Applied Sciences, 2005.

[8] Kilenschmidt, M. Methods for capturing spectrotemporal modulations in automatic speech recognition. *Acta Acustica United With Acustica*, 88(3):416–422, 2002.

[9] Kleinschmidt, M. *Robust Speech Recognition Based on Spectro-Temporal Processing.* PhD thesis, Carl von Ossietzky University, Oldenburg, Germany, 2002.

[10] Kovács, G. and Tóth, L. Phone recognition experiments with 2D DCT spectro-temporal features. In *Proceedings of the International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 143–146. IEEE, 2011.

[11] Kovács, G. and Tóth, L. The joint optimization of spectro-temporal features and neural net classifiers. In *Proceedings of the International Conference on Text, Speech and Dialogue*, pages 552–559. IEEE, 2013.

[12] Kovács, G., Tóth, L., and Compernolle, D. Van. Selection and enhancement of Gabor filters for automatic speech recognition. *International Journal of Speech Technology*, doi: 10.1007/s10772-014-9246-4, 2014.

[13] M., Kleinschmidt and Gelbart, D. Improving word accuracy with Gabor feature extraction. In *Proceedings of the International Conference of Spoken Language Processing (ICSLP)*, pages 25–28. Center for spoken language research, 2002.

[14] Tóth, L. Phone recognition with deep sparse rectifier neural networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6985–6969. IEEE, 2013.

[15] Varga, A. and Steeneken, H. Assessment for automatic speech recognition: Ii. noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems. *Speech Communication*, 12(3):247–251, 1993.

[16] Vesel, K. and M. Karafit, F. Grzl. Convolutive bottleneck network features for LVCSR. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 42–47. IEEE, 2011.

[17] Young, Steve, Evermann, Gunnar, Gales, Mark, Hain, Thomas, Kershaw, Dan, Moore, Gareth, Odell, Julian, Ollason, Dave, Povey, Dan, Valtchev, Valtcho, and Woodland, Philip. *The HTK book version 3.4*. Cambridge University Engineering Department, 2006.