Pattern Recognition Letters

Authorship Confirmation

Please save a copy of this file, complete and upload as the "Confirmation of Authorship" file.

As corresponding author I, György Kovács, hereby confirm on behalf of all authors that:

1. This manuscript, or a large part of it, has not been published, was not, and is not being submitted to any other journal.
2. If presented at or submitted to or published at a conference(s), the conference(s) is (are) identified and substantial justification for re-publication is presented below. A copy of conference paper(s) is(are) uploaded with the manuscript.
3. If the manuscript appears as a preprint anywhere on the web, e.g. arXiv, etc., it is identified below. The preprint should include a statement that the paper is under consideration at Pattern Recognition Letters.
4. All text and graphics, except for those marked with sources, are original works of the authors, and all necessary permissions for publication were secured prior to submission of the manuscript.
5. All authors each made a significant contribution to the research reported and have read and approved the submitted manuscript.

Signature_____ Date_____

List any pre-prints:

Relevant Conference publication(s) (submitted, accepted, or published):

Justification for re-publication:

Pattern Recognition Letters
journal homepage: www.elsevier.com

Increasing the robustness of CNN acoustic models using ARMA spectrogram features and channel dropout

György Kovács[a,**], László Tóth[a], Dirk Van Compernolle[b], Sriram Ganapathy[c]

[a] *MTA-SZTE Research Group on Artificial Intelligence, Tisza Lajos krt. 103, Szeged 6720, Hungary*
[b] *KU Leuven Department of Electrical Engineering (ESAT), Kasteelpark Arenberg 10 postbus 2440, Leuven 3001, Belgium*
[c] *Indian Institute of Science, Department of Electrical Engineering, Javanica Marg, Bengaluru 560012, India*

ABSTRACT

Developing automatic speech recognition systems that are robust to mismatched and noisy channel conditions is a challenging problem, especially when the training and the test conditions are different. Here, we seek to increase the robustness of convolutional neural network (CNN) acoustic models under such circumstances by combining two methods. Firstly, we propose an improved version of input dropout, which exploits the special structure of the input time-frequency representation. Instead of just dropping out random 'pixels' of the spectrogram, the proposed channel dropout approach discards whole spectral channels. We expect that this dropout strategy will force the network to rely less on the whole spectrum, and make it more robust to channel mismatches and narrow-band noise. Secondly, we replaced the standard mel-spectrogram input representation with the autoregressive moving average (ARMA) spectrogram, which was recently shown to outperform the former under mismatched train-test conditions. In our experiments on the Aurora-4 database, the proposed channel dropout method attained relative word error rate reductions of 16% with ARMA features (an absolute improvement of 3%), and 20% with FBANK features (an absolute improvement of 7%) over the baseline CNN, when using the clean training scenario.

1. Introduction

In automatic speech recognition (ASR), convolutional neural network (CNN) acoustic models have been shown to outperform fully connected deep neural nets (DNNs) on various tasks (Abdel-Hamid et al., 2014; Sainath et al., 2015; Tóth, 2015). A recent study found that CNNs have certain advantages over fully connected DNNs in noisy and channel-mismatched testing situations as well (Huang et al., 2015). However, creating robust models remains a challenging problem, especially when we have no samples from the test environment. Here, we present two methods that increase the robustness of CNN acoustic models under such mismatched conditions.

The dropout method was shown to make DNNs generalise better by randomly omitting neurons during training (Hinton et al., 2012), and it is now widely used in ASR (Dahl et al., 2013; Miao and Metze, 2013; Tóth, 2013). When applied to the input features, the original dropout scheme discards or retains each feature independently, assuming that their order does not carry any information. However, in the case of ASR the input is a spectrogram-like time-frequency representation with a well defined structure. Here, we propose a version of input dropout that exploits this, by discarding whole frequency channels, instead of random 'pixels'. We hypothesise that this channel dropout strategy will force the network to rely less on the whole spectrum, making it more robust to channel mismatches and types of noise that affect only certain frequency bands.

Although speech recognition using CNNs and the dropout method are relatively new ideas, our proposal of dropping spectral channels during training is clearly related to some older technologies. One is the so-called multi-band scheme introduced two decades ago (Bourlard and Dupont, 1996). Another related technology is that of data-augmentation (Cui et al., 2014; Ko et al., 2015). We will discuss the connection between these approaches and our solution in Section 2.4.

** Corresponding author: Tel.: +36-62-54-6713;
e-mail: gykovacs@inf.u-szeged.hu (György Kovács)

CNNs seek to exploit the local spectro-temporal correlations of the input, and thus they require a spectrogram-like input representation. Hence, CNNs are usually trained on the log-energy outputs of a mel-scaled filterbank (Abdel-Hamid et al., 2014; Sainath et al., 2015; Tóth, 2015). Our first set of experiments will evaluate channel dropout on the standard mel filterbank features. However, this simple representation clearly has plenty of room for feature engineering. For example, Chang and Morgan experimented with power-normalised spectrum (PNS) features in combination with CNNs (Chang and Morgan, 2014). Recently, Ganapathy proposed an ARMA spectrogram modelling technique that outperformed the PNS features when evaluated with a DNN acoustic model on the Aurora-4 task (Ganapathy, 2015). This ARMA spectrogram representation preserves the local topology of the spectro-temporal feature space, and hence it is a suitable input for CNNs. Our second group of experiments will evaluate the ARMA spectrogram features in combination with CNN acoustic models and channel dropout.

All the experiments described here were performed on the Aurora-4 corpus, which was specially designed to evaluate the noise-robustness of speech recognisers. For this purpose, a clean data set was artificially contaminated with various types of noise. In the 'multi-conditional' training scenario both the training and the test sets contain samples contaminated with the same noise types, so the recogniser has a chance to learn what the noisy recordings are like. In the 'clean' training scenario, however, the recogniser gets only clean training data, which makes this task clearly much more difficult. Although we will evaluate our technology on both the 'multi-conditional' and the 'clean' training scanario, we expect channel dropout to give larger error rate reductions in the case of the latter, where there is a mismatch between the training samples and the noise-contaminated test data.

The paper is organised as follows. In Section 2 we describe our ASR system including the various spectro-temporal input representations and the details of our convolutional network. We then introduce the proposed channel dropout strategy in Section 2.3. In Section 3, we evaluate our CNNs using both the conventional mel-spectrogram features and the ARMA spectrogram features including a sensitivity analysis and tuning of the parameters of input dropout and channel dropout. Finally, in Section 4 we summarise our results and conclusions.

2. Methods

The core of our acoustic model is a CNN that takes a spectro-temporal feature representation as input and produces state posterior estimates as outputs. Fig. 1(a) depicts the convolutional architecture of our CNN, while figures 1(b) and 1(c) show the difference between the standard input dropout where spectro-temporal pixels are randomly dropped during training, versus the proposed dropout scheme where complete frequency channels are dropped, i.e. channel dropout. We will present both the network architecture and the dropout methods in detail, but following a bottom-up presentation, first we introduce the feature extraction methods applied.

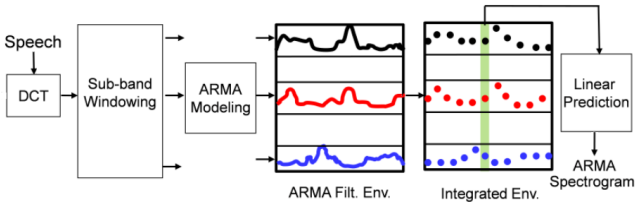

Fig. 2: Block schematic of ARMA feature extraction.

2.1. System description

We use conventional mel filterbank log-energies as a reference input representation for our CNNs, as is common in many other studies (Abdel-Hamid et al., 2014; Sainath et al., 2015; Tóth, 2015). We worked with a mel-filterbank of 42 channels, and the energy values were extended with the corresponding delta and delta-delta features. We will call this feature set the mel-spectrogram or FBANK features.

As an alternative we also use ARMA spectrogram features, which have been shown to have improved robustness in combination with fully connected DNNs (Ganapathy, 2015). Here we give a brief overview of this method, and refer the reader to our earlier study for more detail. The ARMA features are derived using autoregressive moving average (ARMA) spectrogram modelling. The ARMA process is a generalisation of the traditional AR modelling and it can estimate band-pass characteristics, while the AR modelling typically estimates low-pass characteristics (Ganapathy, 2015). The ARMA modelling is applied on the sub-band discrete cosine transform (DCT) components to estimate the temporal envelopes. The ARMA filtered envelopes are then converted into a short-term spectral representation by energy integration. Afterwards, a linear prediction-based spectral smoothing is applied on this spectrogram to get the input features for the CNN.

A block schematic of the ARMA feature extraction scheme is shown in Fig. 2. Let $x[n]$ denote the input signal for $n = 0, \ldots, N-1$ (N is equivalent to 1000 ms in the current study) and let $X[k]$ denote DCT components of the signal $x[n]$. In the proposed framework, we use the DCT components in an ARMA modelling framework to estimate the sub-band envelope. Specifically, the set of coefficients $a_l, l = 1, \ldots, r$ and $b_m, m = 1, \ldots, q$ are estimated such that

$$X[k] = \sum_{l=1}^{r} a_l X[k-l] + \sum_{m=0}^{q} b_m U[k-m], \tag{1}$$

where r, q denote the model order of the AR and MA components and $U[k]$ denotes a zero-mean white noise signal. The AR model is a specific case of the ARMA model with $b_m = 0, \; for \; m > 0$. The ARMA envelope is given by

$$\hat{E}_x[n] = \frac{|\sum_{m=0}^{q} b_m e^{-i2\pi mn}|^2}{|\sum_{l=0}^{r} a_l e^{-i2\pi ln}|^2} \tag{2}$$

The ARMA envelope is the AR envelope (denominator) multiplied by a finite impulse response (FIR) filter provided by the MA modelling (numerator). The MA filter acts as a modulation filter over long temporal regions of signal. Hence, ARMA modelling combines AR estimation with a data-driven modulation filter. Here, we use gain-normalised ARMA envelopes

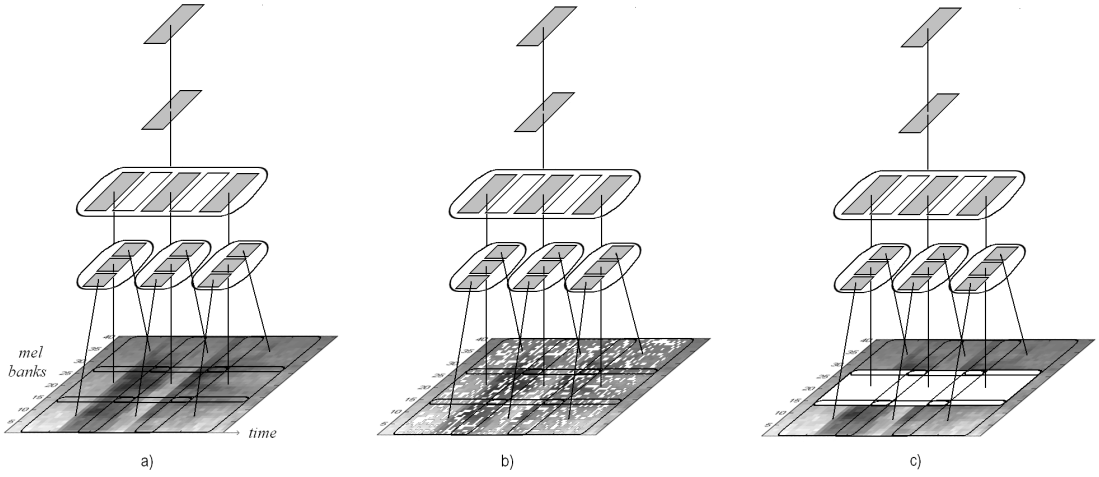

Fig. 1: (a) Illustration of the CNN structure applied here. (b) Input dropout discards randomly selected features (marked by blank dots in the input mel-spectrogram). (c) Channel dropout discards the same amount of features, but this time with an entirely different distribution.

($a_0 = 1$ and $b_0 = 1$). The estimation of ARMA model parameter values is more cumbersome than that of AR modelling and an iterative gradient descent approach is employed (Manolakis et al., 2005). For the ARMA spectrogram estimation, we use $r = 40$, $q = 6$ poles per second per sub-band. We also use a compression factor of 0.2 on the MA part (numerator of Eq. 2) for envelope computation.

Lastly, the ARMA filtered temporal envelopes were processed with spectral smoothing using linear prediction. This is achieved by integrating the sub-band ARMA envelopes in windows of 25 ms duration with a shift of 10 ms. The resulting spectrally smoothed ARMA spectrogram serves as the input for the ASR task. While the previous study used the DCT of the sub-band energy components (computed in 200 ms windows) as input to a DNN model (Ganapathy, 2015), the current framework here uses the ARMA spectrogram directly with a CNN acoustic model.

The resulting ARMA spectrogram representation consisted of 39 spectral bands. In the case of the FBANK representation, the spectral features were extended by adding the delta and delta-delta coefficients, with the aim of representing long-term dependencies of the trajectories. The ARMA features model the temporal envelopes in a much more sophisticated way; that is, the MA filter in the ARMA model performs data driven modulation filtering, similar to the conventional delta coefficients. Because of this, extending the ARMA features with the usual delta coefficients is pointless. However, as the ARMA-based spectral envelopes are smooth in nature, we found that adding the *spectral* deltas enhances the variations across the frequency axis, which improved the results in recognition experiments. Hence, the 39 ARMA features were extended by adding 39 derivative-like features obtained as the difference between neighbouring bands ($band(K + 1) - band(K - 1)$).

2.2. The convolutional network structure

The hierarchical convolutional network structure used in this study (Fig. 1) is essentially the same as the one that yielded excellent results in a previous study with the TIMIT database (Tóth, 2014, 2015). We first summarise the main properties of the network when used with the FBANK features; full details are found in the above references. The lowest layer of the network divides the frequency axis into 9 wider frequency channels (the figure shows only 3 channels for clarity), and processes these using separate groups of neurons or 'filters'. These filters process windows of 9 frames times 9 mel bands (this size was found optimal in our earlier study (Kovács et al., 2015)). Although we found earlier that initialising the filters with Gabor filter coefficients is slightly better than random initialisation (Kovács et al., 2015; Chang and Morgan, 2014), for the sake of simplicity, here we used 9 randomly initialised filters per channel (with the delta and delta-delta maps having their own dedicated filters). At this point, the model may perform frequency-domain convolution by evaluating and pooling the filters at slightly shifted frequency positions. The spectro-temporal features extracted by the filters are concatenated and processed by three additional, fully connected layers (for clarity, only one of these is shown in the figure). These contain 1400-1400-400 maxout neurons (Cai et al., 2013; Miao et al., 2013). This lower part of the network is evaluated at nine positions along the time axis (Fig. 1(a) depicts three of these) with three frames being skipped (i.e., down-sampling). The resulting vectors are concatenated and processed by two fully connected layers with 1400-1400 neurons, while the output layer is a softmax layer with 1998 units. This hierarchical architecture markedly increases the time-span of the model without increasing the number of weights. While Veselý et al. interpreted this solution as a time-domain convolution (Veselý et al., 2011), we called this network structure the hierarchical convolutional model (Tóth, 2015), in order to avoid confusion with CNNs that perform convolution only along the frequency axis (Abdel-Hamid et al., 2014; Sainath et al., 2015). More recently, Peddinti et al. showed that this architecture is essentially a time-delay neural network (Waibel et al., 1989) extended with sub-sampling (Peddinti et al., 2015).

We made some small modifications to our CNN for the ARMA spectrogram feature input. In particular, as the ARMA spectrogram had fewer bands than the mel-spectrogram, we used 8 convolutional channels instead of 9. We also decreased

the number of positions the lower part of our CNN was evaluated on to five (in order to cover approximately the same range in the time domain). Another difference requiring modification was that instead of having two sets of derivative features, the ARMA representation had only one. While the network was adjusted to these slight changes in the input, the main structure and size of the network (i.e., size of the fully connected layers) was left unaltered.

2.3. Channel dropout

The standard dropout method randomly omits neurons from the network during training (Hinton et al., 2012). This improves the generalisation abilities of the network, and it is now commonly agreed that applying dropout to the hidden layers of DNNs is beneficial to ASR performance (see, e.g., Dahl et al. (2013); Miao and Metze (2013)). The original dropout study reported additional improvements by applying dropout to the input features as well (Hinton et al., 2012). However, in the framework of ASR, Deng et al. found that "applying dropout to input filterbank features has not been effective" (Deng et al., 2013). Miao and Metze reported that "an input dropout factor greater than 0 definitely degrades the recognition results" (Miao and Metze, 2013). Hence the usefulness of input dropout in ASR is questionable, and we will run experiments to see how it performs with our input features and CNN architecture. In the tables we will simply refer to the dropout method that randomly drops input features as the 'standard' input dropout.

Here, we propose a modified version of input dropout. The original dropout strategy selects the input components to be discarded in a random fashion. As here we have a spectrogram-like input, the distribution of dropped pixels should look similar to how the addition of white noise would contaminate the input (see our illustration in Fig. 1(b)). However, real-life background noise is more likely to affect the spectrogram in patches, rather than spreading all over the time-frequency plane in a uniform manner. Furthermore, it is known that humans can recognise bandpass-filtered speech surprisingly well (Allen, 1994). These observations motivated us to modify the dropout strategy so that it eliminates full frequency channels instead of just pixels from the spectral map. The frequency channels we drop consist of several neighbouring mel-filter channels, so that they coincide with the wider frequency bands covered by the convolutional filters of the CNN. This way, dropping one channel means that the corresponding set of the convolutional filters receives zero input, while the operation of the others is left untouched. Consequently, the layer that merges the output of the filters is forced to learn that it should not rely on all of the channels. Figures 1(b) and 1(c) show the difference between standard input dropout and channel dropout (for the case of 3 processing channels). Clearly, with the proper selection of the dropout rate parameters, the two methods discard the same amount of data points, but with a quite different distribution.

Another difference is that while the original method generates a new dropout mask for each data instance, here we use the same dropout mask within a given mini-batch. The advantage of this method is that it allows a faster form of matrix multiplication (Graham et al., 2015). Moreover, it can be easily and efficiently combined with a CNN, because we can implement the dropout of a convolutional channel for the whole mini-batch by simply skipping the evaluation of the given filters and replacing their output by zeros. Although Graham et al. thoroughly evaluated their batch-wise dropout method and concluded that it gives basically equivalent results with the standard frame-wise solution, we will also compare the two approaches in the tables.

2.4. Relation to Prior Work

Channel dropout forces the network to make decisions based on a subset of spectral channels, and thus rely less on the whole spectrum. We expect that this will make the network more robust to channel mismatches and types of noise that affect only certain frequency bands. The multi-band model, introduced two decades ago, was based on similar motivations. However, the training of deep structures was an open question, so separate classifiers were trained on the spectral bands (or, subsets of bands). The band-based estimates were combined by a sophisticated decision logic which, in the simplest case, consisted of a 'merger' neural network (Bourlard and Dupont, 1996). In a way, channel dropout can be viewed as a modern version of band-based training adjusted to deep convolutional networks. In fact, in a recent paper Mallidi et al. proposed a multi-stream combination method that drops streams during the training of the merger network (Mallidi and Heřmanský, 2016). While this solution is clearly related to ours, there are significant differences both in the motivation and the implementation details. As is usual with the multi-band scheme, they train a large set of DNNs instead of just one deep structure, and at test time a performance monitor is applied to decide which streams should be kept. The unselected streams are replaced by zeros, and their motivation for training the merger network with dropped streams is to prepare the merger net for the possibility of channels being zeroed out by the stream selection process. Here, we train just one deep convolutional network without any explicit channel selection process, and our motivation for dropout is the same as that for standard input dropout (Hinton et al., 2012). Hence, while the old multi-band systems were based only on a crude perceptual motivation, our approach is also supported by the machine learning theory of dropout. Lastly, while the solution of Mallidi et al. is more complicated to implement, and also slower at run time (as the performance monitor needs to be invoked), the results we present here are significantly better than those given in their study (Mallidi and Heřmanský, 2016).

Data augmentation is an approach for increasing the robustness of DNNs by artificially generating additional training vectors from the existing ones (Cui et al., 2014; Ko et al., 2015). Recently, in a study by Bouthillier et al. it was pointed out that dropout can indeed be interpreted as a kind of data augmentation (Bouthillier et al., 2015). Hence, one may also regard channel dropout as a special case of data augmentation where, by randomly deleting spectral bands, we artificially extend our set of training data with additional distorted versions of the training vectors on the fly. According to this interpretation, while Ko et al. modified the speech signals in the time domain (Ko et al., 2015), channel dropout belongs to the family of data augmentation techniques that manipulate the data in the spectral domain (Cui et al., 2014).

Table 1: *The effect of channel dropout on the frame error rates for the train and development sets, and on the word error rates for the test set (using the FBANK features and the multi-conditional training set). The baseline (no dropout) score is given in the $P = 0$ column.*

Frame error rate (%) – Train					
N \ P	*0*	*0.2*	*0.4*	*0.6*	*0.8*
5	30.2	32.9	34.0	34.9	37.4
6		33.1	35.2	36.3	39.1
7		34.2	37.1	38.3	41.0

Frame error rate (%) – Dev				
0	*0.2*	*0.4*	*0.6*	*0.8*
36.8	36.9	36.7	36.8	37.7
	36.8	36.9	36.9	37.9
	37.0	37.3	37.1	37.9

Word error rate (%) – Test				
0	*0.2*	*0.4*	*0.6*	*0.8*
11.6	11.3	11.2	10.9	11.3
	11.2	11.1	**10.8**	11.2
	11.2	11.2	10.9	11.1

3. Results

3.1. Experimental Setup

We evaluated the proposed method on the Aurora-4 data base (Hirsch and Pearce, 2000). For testing purposes, a test set of 330 utterances was used in 14 different versions (clean/noise contaminated speech recorded with a Sennheiser close-talking microphone, or with one of the several secondary microphones). Results are often reported on the indvidual test sets, or on bigger sets created by averaging the results on some of the individual sets: test set A (clean recordings with the Sennheiser microphone), set B (6 noise-corrupted versions of set A), set C (clean recordings with a secondary microphone), and set D (6 noise-corrupted versions of set C). The database contains two training sets, namely the clean set and the multi-condition set, both consisting of 7138 utterances. The multi-condition set contains samples from the secondary microphones and the various types of noisy conditions, while the clean training set consists of only the clean training data recorded with the Sennheiser microphone.

First, we trained a HMM/GMM model with Kaldi's Aurora-4 recipe. Then, we performed forced alignment with this model, and utilised the acquired frame-level state labels as training targets for our in-house CNN implementation that we used to replace Kaldi's DNN. Our CNN was trained with backpropagation using the frame-level cross-entropy error function. A random 10% of the training set was held out as the development set used for the early stopping of training, and for tuning the meta-parameters. Lastly, the decoding and scoring steps were again performed with Kaldi scripts, using the standard trigram language model and the 5k word vocabulary.

The channel dropout method was parametrised by using two variables. With parameter P we can tune the probability of the channel dropout being applied to the current batch of data. The other parameter, N sets the maximum number of channels that can be discarded by channel dropout (that is, for the test configuration used here the value of N is between 1 and 9). If the current batch is selected for dropout (according to P), then a random number q is generated in the $[1, N]$ range, and we discard the input data in q randomly chosen convolutional channels. Although dropout slows down the convergence of the training process, so one can perform more sweeps through the training data when using dropout (Dahl et al., 2013), here we did not modify the training epochs or the stopping criterion. This way our dropout results might be slightly suboptimal, but the training times stayed roughly the same as those without dropout.

3.2. Results with mel-spectrogram features

We first evaluated the usefulness of channel dropout using FBANK features in the multi-condition training scenario. As our training process optimised the frame-level cross-entropy, let us first examine the frame-level error rates attained with different parameter settings (listed in Table 1, with the baseline obtained without channel dropout in the column of $P = 0$). As one would expect, channel dropout makes the learning process more difficult, so the frame-level error rate on the train set quickly increases when we increase either P or N. However, examining the frame error rates on the development set, we see that we can go up to $P = 0.6$ and $N = 6$ with the error remaining roughly the same as that for the baseline. As narrowing the gap between the training and development error rates decreases the chance of overfitting the training data, a reasonable heuristic for meta-parameter tuning is to chose the largest P and N values for which the error on the development set is not significantly larger than that for the baseline model. According to this strategy, Table 1 suggests choosing $P = 0.6$ and $N = 6$.

Let us now examine the word-level error rates on the test set (the rightmost table in Table 1). The scores are quite consistent with the frame-level errors obtained on the development set. Actually, the word-level error rates are lower than that for the baseline for all parameter values, and are quite stable with respect to P and N. While the WER is the lowest (10.8%) at parameter values suggested by our heuristic ($P = 0.6$, $N = 6$), slightly different P and N values result in similarly low WER scores. The relative word error rate reduction[1] compared to not using channel dropout is 7.3% (an absolute error rate reduction of 0.9%), which was found to be statistically significant in a paired t-test with $p = 0.00008$. The score of 10.8% also compares favourably with results reported by other authors: for example, the Kaldi recipe gives 13.6% with a DNN, while Huang et al. got 13.4% using FBANK features and a CNN (Huang et al., 2015). It should be acknowledged that better results have been obtained (see, for example, Qian et al. (Qian et al., 2015)), but these methods rely on explicit noise and/or speaker adaptation, which is not required in our approach.

The second column of Table 2 compares the performance of channel dropout with two versions (frame-wise and batch-wise) of standard input dropout. First, we sought the optimal dropout probability value for standard (frame-wise) input dropout. The parameter values of $N = 6$ and $P = 0.6$ mean that on average

[1] The relative error rate reduction is a standard error metric used to measure the improvement in automatic speech recognition. Given an earlier error rate ($ERR1$), and a new error rate ($ERR2$), it is computed by the following formula: $100 \cdot (ERR1 - ERR2)/ERR1$.

Table 2: *The word error rates obtained with various versions of input dropout, using the FBANK features in the multi-condition and clean training scenarios.*

Method	Training scenario	
	Multi-condition	Clean
channel dropout ($P = 0.6, N = 6$)	10.8%	26.8%
standard input dropout ($P = 0.1$)	11.3%	31.4%
standard input dropout ($P = 0.2$)	11.0%	31.4%
standard input dropout ($P = 0.3$)	11.4%	33.3%
batch-wise input dropout ($P = 0.2$)	10.8%	30.5%
no dropout	11.6%	33.7%

Table 3: *The word error rates obtained with various versions of input dropout, using the ARMA features in the multi-condition and clean training scenarios.*

Method	Training scenario	
	Multi-condition	Clean
channel dropout ($P = 0.6, N = 6$)	10.8%	16.0%
channel dropout ($P = 0.4, N = 5$)	10.5%	16.5%
standard input dropout ($P = 0.1$)	11.1%	19.2%
standard input dropout ($P = 0.2$)	11.4%	19.9%
batch-wise input dropout ($P = 0.1$)	11.1%	19.0%
no dropout	10.7%	19.1%

3.5 channels (out of 9) are dropped with a probability of 0.6, which corresponds to a $(3.5/9) \cdot 0.6 = 0.23$ equivalent input dropout percentage. In comparison, we evaluated conventional input dropout with dropout rates $P = 0.1, 0.2, 0.3$. The best result was obtained at $P = 0.2$, where the word error rate was 11.0%. Next, we repeated the evaluation at $P = 0.2$ using the batch-wise version of dropout. In this case, we got a slightly better score of 10.8%. While the scores obtained using dropout (10.8%, 11.0% and 10.8%) proved to be significantly better than the baseline score obtained with no dropout ($p < 0.0007$), these scores were not significantly different from each other. Hence, in this experiment the channel dropout method turned out to be useful, but no better than standard dropout.

Next, we examined how channel dropout behaves when only clean training data is used. To test the robustness of our parameter selection strategy, we did not repeat the process of tuning, but worked with those parameters found optimal previously ($P = 0.6, N = 6$). The last column of Table 2 shows the baseline word error rate obtained without channel dropout, and the score got with channel dropout. By way of comparison, Huang et al. reported 28.9% with their CNN (Huang et al., 2015). In comparison, we evaluated conventional input dropout with dropout rates of $P = 0.1, 0.2, 0.3$ (see the third column of Table 2). The optimal performance was got at 0.1, with a word error rate of 31.4%. This shows that while input dropout reduced the error rate by 6.8%, channel dropout decreased it by a much larger amount of 20.4%. In this case, channel dropout was significantly better compared both to the baseline and to the standard input dropout ($p < 0.000001$). This accords with our expectation that channel dropout is the most beneficial when we have no noisy training samples, so the network cannot adapt to the noise using just the training data. Lastly, we repeated the evaluation of standard dropout with $P = 0.2$ in a batch-wise manner. The 30.5% we obtained is lower than the 31.4% of frame-wise dropout, but much higher than the 26.8% of channel dropout.

Here, we should mention that the channel dropout scheme heavily exploits the special structure of CNNs; that is, the fact that the channels discarded coincide with the processing channels of the CNN. For a fully connected DNN it would make no difference whether the dropped spectral points fall within the same channel or not, so channel dropout would perform in similar way to standard input dropout. The performance gain we got comes from the fact that both the model and the dropout scheme are adjusted to the band-based topology of the input.

3.3. Results with the ARMA spectrogram features

We first evaluated the ARMA features under the multi-conditional training scenario. As the second column of Table 3 shows, compared to the FBANK baseline score of 11.6%, the introduction of the ARMA features reduced the error by 7.8% (an absolute improvement of 0.9%), even without using channel dropout. However, turning on channel dropout with the parameter values of $P = 0.6$ and $N = 6$ did not reduce the error rates any further. Then, we looked for the parameter values that would give the best result on the test set. The 10.5% we got with $P = 0.4$ and $N = 5$ is just slightly, but not significantly better than the baseline 10.7%. We also evaluated the standard input dropout with $P = 0.1$ and 0.2, as well as the batch-wise input dropout with $P = 0.1$, but we got worse results than the baseline. Altogether, in contrast with the FBANK feature set, in the multi-conditional training scenario neither dropout method helped. We assume that the ARMA technique can already efficiently handle the noise when we have training samples from all the noisy conditions, but this would require a deeper analysis.

In the final set of experiments we used the ARMA features and only clean training data. Comparing the FBANK result under clean conditions (33.7% in Table 2) with those of the ARMA feature set (19.1% in Table 3), we see that employing the latter yields an impressive 43.3% relative error rate reduction (an absolute improvement of 14.6%). The improvement was similar in our earlier study, where we evaluated the ARMA features with fully connected DNNs (Ganapathy, 2015). Next, we turned on channel dropout with $P = 0.6$ and $N = 6$, and attained a word error rate of 16.0%. This means that while channel dropout did not help the ARMA features under the multi-conditional training scenario, here we got a 16.2% relative error rate reduction (an absolute improvement of 3.1%) compared to 19.1% using the ARMA features, but no dropout (the improvement is significant at $p < 0.0001$). Here, we also tried to vary P between 0.4-0.8 and N between 4-6, and even the worst result we obtained was 16.5%, proving channel dropout to be quite stable across a wide range of parameter values. Lastly, we evaluated standard input dropout at two P values (and batch-wise input dropout at one P value) and, similar to the multi-conditional case, we got worse results than the baseline. In summary, while channel dropout performed no better than standard input dropout in the multi-conditional training scenario, it significantly outperformed it when only clean training data was used. This justifies our expectations that channel dropout increases the generalisation capability of the model under mismatched training and test conditions.

Table 4: *The performance of the CNN when operating on FBANK and ARMA features without, and with, channel dropout, using only the clean training set.*

Data set		FBANK				ARMA			
		no dropout	channel dropout	absolute improvement	relative improvement	no dropout	channel dropout	absolute improvement	relative improvement
Set A	Clean	3.8%	3.1%	0.7%	17.8%	3.7%	3.8%	-0.2%	-4.6%
Set B	*Car*	13.0%	7.2%	5.7%	44.2%	6.1%	5.5%	0.6%	9.5%
	Babble	20.1%	15.7%	4.4%	21.7%	13.9%	11.0%	2.8%	20.4%
	Restaurant	29.3%	24.2%	5.0%	17.2%	18.1%	16.9%	1.2%	6.6%
	Street	30.7%	21.0%	9.6%	31.4%	13.9%	12.1%	1.8%	13.3%
	Airport	20.7%	15.2%	5.5%	26.5%	13.3%	12.6%	0.7%	5.2%
	Train	28.2%	20.2%	8.0%	28.4%	15.1%	12.4%	2.7%	17.7%
	Average	23.7%	17.3%	6.4%	27.0%	13.4%	11.8%	1.6%	12.2%
Set C	Clean	35.7%	29.5%	6.2%	17.3%	13.6%	9.4%	4.2%	30.9%
Set D	*Car*	43.1%	35.7%	7.4%	17.2%	21.1%	15.3%	5.8%	27.5%
	Babble	46.3%	39.0%	7.3%	15.7%	29.9%	24.1%	5.8%	19.4%
	Restaurant	48.9%	40.2%	8.6%	17.7%	31.0%	27.6%	3.5%	11.2%
	Street	55.0%	44.5%	10.5%	19.1%	28.8%	24.6%	4.2%	14.6%
	Airport	45.6%	38.3%	7.3%	16.0%	29.0%	24.3%	4.7%	16.1%
	Train	51.3%	41.3%	10.0%	19.5%	29.6%	24.2%	5.4%	18.2%
	Average	48.4%	39.8%	8.5%	17.6%	28.2%	23.3%	4.9%	17.3%
Average		33.7%	26.8%	6.9%	20.4%	19.1%	16.0%	3.1%	16.2%

The word error rates of 26.8% and 16.0% reported in tables 2 and 3, respectively, are average scores over the four test sets. To get a deeper insight into the precise conditions where channel dropout is the most beneficial, we performed a detailed analysis of how the error rate improvement varies under different testing conditions. Table 4 lists the word error rates got for each test set, and for each noise type (within sets B and D). The first thing we notice is that under ideal conditions when there is neither channel distortion nor additive noise present (i.e., set A), channel dropout only improves the recognition accuracy in the case of FBANK features, while with ARMA features, it actually slightly degrades the performance. However, for test set B, when additive noise is present, channel dropout yields a significant gain of 12.2% relative error rate reduction (an absolute improvement of 1.2%) on ARMA features, and a 27.0% relative error rate reduction (an absolute improvement of 6.4%) on FBANK features, which represents the biggest relative gain for the feature set in question. The biggest relative improvement on the ARMA features (30.9%, that is an absolute improvement of 4.2%) however is obtained on set C, when the microphone transfer characteristics are different, but no noise is involved. The improvement obtained on the same set with FBANK features is 17.3% (an absolute improvement of 6.2%), almost identical to the relative improvements attained on set D with either feature sets. These scores accord with our hypothesis that channel dropout will be beneficial under mismatched training and test conditions, in particular for mismatched transfer channel characteristics, and in the presence of additive noise. The effect of channel dropout in the latter case, however, is mitigated when used on ARMA features. Again, this might be due to the noise robustness of ARMA features.

Lastly, in Table 5, we compare our best score with scores obatined using other noise robust methods, as well as our earlier results, and some recent results found in the literature. Our first base of comparison is a standard DNN with 7 hidden layers, each consisting of 2000 neurons, using only the ARMA features (and their derivatives) as input. We then applied the Gabor filter set introduced for noise robustness in (Kovács et al., 2015) on the ARMA features, plus the Δ coefficients. We used the resulting features as an input to another seven layer DNN. The results of these experiments are listed in Table 5. As can be seen, while using the Gabor filter set improves the performance, the WER scores attained only come close to those attained with our CNN baseline, and are much higher than those attained with our proposed channel dropout method.

Next, we compared the result attained here using ARMA features with our earlier results attained using the same features. One base of comparison is our earlier paper where the ARMA features were evaluated using a fully connected DNN (Ganapathy, 2015). The results with this model were slightly better than our baseline here, which was perhaps due to the DCT postprocessing of the feature trajectories before feeding them into the DNN. However, our CNN with channel dropout significantly outperformed this score ($p < 0.0007$). We see similar results when we compare the WER scores of this study with those we attained by applying the multi-band approach on the ARMA features (Kovács and Tóth, 2016). Our multi-band model applied a quite complex network: 4 convolutional neural nets were trained, each using two channels of the ARMA representation (and two from the corresponding delta-like map). Each of these CNNs had 4 filtering layers with 9 filters, a convolutional layer with 200 neurons, two further layers with 1000 neurons each, and in front of the output layer of 1997 neurons, a bottleneck layer with 50 neurons. The results of the four convolutional neural nets trained independently were merged by a fifth neural net, which contained two hidden layers, each consisting of 1000 neurons. The input of this merger DNN was provided

Table 5: *Comparing the result of using channel dropout with ARMA features with results of using fully connected DNNs, results of our earlier experiments, and with results in the recent literature, when using the clean training set.*

Method	WER
CNN with ARMA features, channel dropout	**16.0%**
DNN with ARMA features, no dropout	19.7%
DNN with ARMA features plus Gabor features and Δ coefficients	19.2%
DNN with ARMA features plus DCT (Ganapathy, 2015)	18.5%
Multi-band processing using CNN with ARMA features (Kovács and Tóth, 2016)	17.8%
DNN with DNN speech enhancement of FBANK (Jun et al., 2014)	17.5%
DNN with Spectral masking (Li and Sim, 2013)	22.8%
CNN with PNS features plus Gabor Filter Kernels (Chang and Morgan, 2014)	22.9%
DNN with Exemplar Based Enhancement (Baby et al., 2015)	26.8%
CNN with FBANK features (Huang et al., 2015)	28.9%

by the output of the bottleneck layers in the four CNNs. We notice in Table 5 that our WER scores with multi-band processing – although lower than those achieved by our baseline and those reported in (Ganapathy, 2015) – are significantly higher than those attained using the channel dropout method proposed here ($p < 0.00005$).

The remaining rows of Table 5 list recent results from the literature, all obtained with a DNN or a CNN (as there is now common agreement that these methods outperform conventional HMM/GMMs). While Huang et al. apply a simple CNN on the FBANK features, others apply a sophisticated feature extraction method (Baby et al., 2015; Li and Sim, 2013) or a refined CNN architecture (Chang and Morgan, 2014). We see that the ARMA feature set outperforms both these approaches by a good margin. And when combined with channel dropout, it outperforms the results of Jun et al. even though in their study the front-end utilises samples from the multi-condition training set. The reason for this is that the training phase of their speech enhancement DNN requires pairwise clean-noisy data, which they got from the multi-condition training set (Jun et al., 2014).

4. Conclusions

Here, we introduced channel dropout as a novel input dropout method, which is highly beneficial in combination with CNN based acoustic modelling. The effectiveness of this method was demonstrated using the AURORA-4 database with different input representations (filterbank features and ARMA spectrogram features) and for different scenarios (multi-condition training and mismatched train/test). Moreover, implementing channel dropout is straightforward and requires a negligible additional computational cost.

Using the multi-condition training set and filterbank features, channel dropout yielded a relative error rate reduction of 7.3% (an absolute improvement of 0.9%) vs. no dropout. More dramatic improvements were obtained in the case of train-test mismatch conditions. When we just use the clean training data, for filterbank features the channel dropout method yielded a relative error rate reduction of 20.4% over the no-dropout case, and 12.3% over batch-wise input dropout, while for ARMA features it yielded a relative error rate reduction of approximately 16% over both no dropout and standard input dropout (absolute improvements of 6.9%, 3.7% and approximately 3%, respectively). Overall, we can say that channel dropout beats standard input dropout by a large margin. Furthermore, with ARMA features our system produced an absolute error rate of 16.0%, which is among the best results published for this task.

References

Abdel-Hamid, O., Mohamed, A.R., Jiang, H., Deng, L., Penn, G., Yu, D., 2014. Convolutional neural networks for speech recognition. IEEE/ACM Trans. ASLP 22, 1533–1545.

Allen, J.B., 1994. How do humans process and recognize speech? IEEE Trans. on Speech and Audio Proc. 2, 567–577.

Baby, D., Gemmeke, J.F., Virtanen, T., Van Hamme, H., 2015. Exemplar-based speech enhancement for deep neural network based automatic speech recognition, in: Proc. ICASSP, pp. 4485–4489.

Bourlard, H., Dupont, S., 1996. A new ASR approach based on independent processing and recombination of partial frequency bands, in: Proc. ICSLP, pp. 426–429.

Bouthillier, X., Konda, K., Vincent, P., Memisevic, R., 2015. Dropout as data augmentation. ArXiv e-prints `arXiv:1506.08700`.

Cai, M., Shi, Y., Liu, J., 2013. Deep maxout neural networks for speech recognition, in: Proc. ASRU, pp. 291–296.

Chang, S.Y., Morgan, N., 2014. Robust CNN-based speech recognition with Gabor filter kernels, in: Proc. Interspeech, pp. 905–909.

Cui, X., Goel, V., Kingsbury, B., 2014. Data augmentation for deep neural network acoustic modeling, in: Proc. ICASSP, pp. 5619–5623.

Dahl, G.E., Sainath, T.N., Hinton, G.E., 2013. Improving deep neural networks for LVCSR using rectified linear units and dropout, in: Proc. ICASSP, pp. 8609–8613.

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

Ganapathy, S., 2015. Robust speech processing using ARMA spectrogram models, in: Proc. ICASSP, pp. 5029–5033.

Graham, B., Reizenstein, J., Robinson, L., 2015. Efficient batchwise dropout training using submatrices. ArXiv e-prints `arXiv:1502.02478`.

Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. CoRR abs/1207.0580.

Hirsch, H.G., Pearce, D., 2000. The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions, in: Proc. ASR2000, pp. 29–32.

Huang, J.T., Li, J., Gong, Y., 2015. An analysis of convolutional neural networks for speech recognition, in: Proc. ICASSP, pp. 4989–4993.

Jun, D., Qing, W., Tian, G., Yong, X., Li-Rong, D., Chin-Hui, L., 2014. Robust speech recognition with speech enhanced deep neural networks, in: Proc. Interspeech, pp. 616–620.

Ko, T., Peddinti, V., Povey, D., Khudanpur, S., 2015. Audio augmentation for speech recognition, in: Proc. Interspeech, pp. 3586–3589.

Kovács, Gy., Tóth, L., 2016. Multi-band noise robust speech recognition using deep neural nets (in Hungarian), in: Proc. MSZNY, pp. 287–294.

Kovács, Gy., Tóth, L., Van Compernolle, D., 2015. Selection and enhancement of Gabor filters for automatic speech recognition. Int. Journal of Speech Tecnhology 18, 1–16.

Li, B., Sim, K.C., 2013. Improving robustness of deep neural networks via spectral masking for automatic speech recognition, in: Proc. ASRU, pp. 279–284.

Mallidi, S.H., Heřmanský, H., 2016. Novel neural network based fusion for multistream ASR, in: Proc. ICASSP, pp. 5680–5684.

Manolakis, D.G., Ingle, V.K., Kogon, S.M., 2005. Statistical and adaptive signal processing: spectral estimation, signal modeling, adaptive filtering, and array processing. Norwood: Artech House.

Miao, Y., Metze, F., 2013. Improving low-resource CD-DNN-HMM using dropout and multilingual DNN training, in: Proc. Interspeech, pp. 2237–2241.

Miao, Y., Metze, F., Rawat, S., 2013. Deep maxout networks for low-resource speech recognition, in: Proc. ASRU, pp. 398–403.

Peddinti, V., Povey, D., Khudanpur, S., 2015. A time delay neural network architecture for efficient modeling of long temporal contexts, in: Proc. Interspeech, pp. 3214–3218.

Qian, Y., Yin, M., You, Y., Yu, K., 2015. Multi-task joint learning of deep neural networks for robust speech recognition, in: Proc. ASRU, pp. 310–316.

Sainath, T.N., Kingsbury, B., Saon, G., Soltau, H., Mohamed, A., Dahl, G., Ramabhadran, B., 2015. Deep convolutional neural networks for large-scale speech tasks. Neural Networks 64, 39–48.

Tóth, L., 2013. Convolutional deep rectifier neural nets for phone recognition, in: Proc. Interspeech, pp. 1722–1726.

Tóth, L., 2014. Convolutional deep maxout networks for phone recognition, in: Proc. Interspeech, pp. 1078–1082.

Tóth, L., 2015. Phone recognition with hierarchical convolutional deep maxout networks. EURASIP Journal on Audio, Speech and Music Processing 25.

Veselý, K., Karafiát, M., Grézl, F., 2011. Convolutive bottleneck network features for LVCSR, in: Proc. ASRU, pp. 42 – 47.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K., 1989. Phoneme recognition using time-delay neural networks. IEEE Trans. ASSP 37, 328–339.