Multifeature Modular Deep Neural Network Acoustic Models

Kevin Kilgour, Alex Waibel

International Center for Advanced Communication Technologies - InterACT,
Institute of Anthropomatics and Robotics
Karlsruhe Institute of Technology (KIT), Germany
kevin.kilgour,alexander.waibel@kit.edu

Abstract

This paper presents and examines multifeature modular deep neural network acoustic models. The proposed setup uses well trained bottleneck networks to extract features from multiple combinations of input features and combines them using a classification deep neural network (DNN).

The effectiveness of each feature combination is evaluated empirically on multiple test sets for both a classical DNN as well as a for modular DNNs using only a single module. Modular DNNs using two or more modules are shown to reduce the WER by up to 11.5% relatively compared to a baseline DNN and give the best overall performance on both test sets.

1. Introduction

The first step in speech recognition is to extract a stream of feature vectors form the audio. Although many of these so called front-ends are fundamentally similar and equally useful, they are still, to some extent, complementary and the outputs of ASR systems trained separately on different front-ends can be combined in such a manner that the combined output contains fewer transcription errors than either of the individual outputs [1, 2]. While very useful, this high level combination method has the disadvantage of requiring multiple ASR systems to be run in parallel.

In this paper an alternative approach is proposed that uses modular deep neural networks (mDNNs [3]) to combine the features in a single acoustic model. An mDNN can be seen as an extension of a time-delay neural network (TDNN) [4]. TDNNs are designed to be time invariant and work on sequences of feature vectors. As well as the current feature vector x_t the neurons of the first hidden layer are also connected to a few of the preceding feature vectors $x_{t-1}x_{t-2}, ...$ The *time-delay* procedure is applied at the transition from the first hidden layer to the second hidden layer. The neurons of the second hidden layer also possess connections to the first hidden layer's outputs at the preceding steps.

While both TDNNs [5] and CNNs [6, 7] may have many *time-delayed* or convolutional layers these layers are normally directly connected to their preceding layers. Modular DNNs on the other hand use well trained deep neural networks to connect the input layer to the *time-delayed* layer. As these network modules are trained as bottleneck features (BNF) [8] we refer to this layer as the bottleneck layer. In this paper we show how using multiple features as inputs to the BNF modules can improve the performance of an mDNN and go on to experiment with using an mDNN to combine multiple different BNF modules.

This paper is structured as follows: after an overview of the relevant related work in section 2 a multifeature DNN AM is introduced as well as all the features used throughout this paper. This is followed in section 3 by a description of the proposed multifeature DNN. Section 5 explains how the neural networks are evaluated the presents and results after which section 6 concludes the paper with a short summary.

2. Related Work

A method of using BNFs to combine multiple feature streams proposed in [9], shows that combining MFCC, PLP and gammatone features in the input layer of an MLP can lead to a system that performs better than the system combination of the lattices of the individual systems. The MLP using the combined input feature also outperforms the best single feature MLP by a small amount. In contrast to this work they, however, only look at shallow networks. Instead the authers later focus on integrating the multiple features into a shallow RNN [10] trained to classify the phone targets. Stacking MFCCs and MVDRs (Minimum Variance Distortionless Response) at the input of a DNN was also found to be helpful in bottleneck feature extraction for German Broadcast News [11] as well as for the NIST 2013 OpenKWS evaluation [12].

In [13] various tonal models and methods of integrating tonal features are analyzed on both tonal and non tonal languages. That work reports, for the first time, results of using fundamental frequency variation [14] features for speech recognition of tonal languages and finds that early integrating tonal features consistently leads to a reduction in WER even on non-tonal languages.

A version of the modular DNN designed for low resource languages is discussed in [3] and modified to make use of language resources outside of the target language [15].

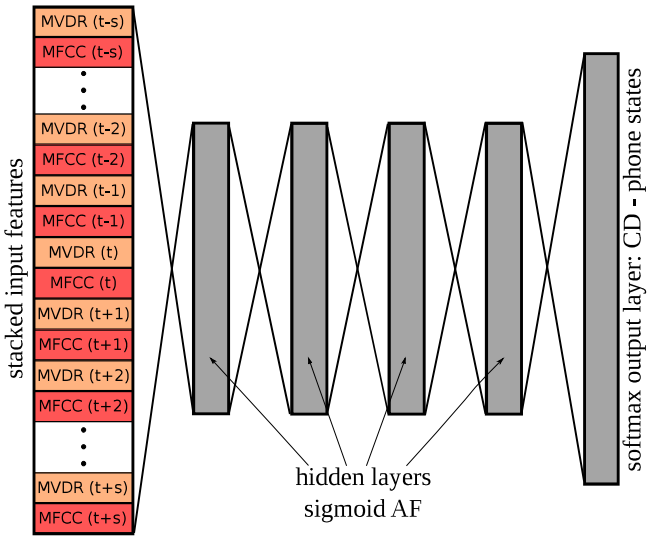

Figure 1: *An example of a deep neural network with 4 hidden layers and an output layer with corresponding CD phone states. Its input is a 2s+1 frame window containing both MFCC and MVDR features.*

3. Multifeature Deep Neural Network Accoustic Models

Multifeature DNN-AMs are feed forward neural networks that merge multiple different input feature in the input layer of the neural network. An example DNN AM is shown in figure 1. It has an input layer that uses stacked MVDR+MFCC features in a window spanning from s frames prior to the current frame to s frames after the current time frame, followed by four hidden layers that use the sigmoid activation function and a softmax output layer where the neurons correspond to the context dependent phone states. The example contains two of the four different input feature used in this paper:

- **Mel Frequency Cepstral Coefficients (MFCC)**: MFCCs have established themselves as the most common front-end feature in speech recognition. They are computed by applying a discrete cosine transformation to log Mel features.
- **Minimum Variance Distortionless Response (MVDR) Spectrum**: MVDR [16] features are an improvment on basic linear prediction features [17]. In some circumstances warped Minimum Variance Distortionless Response (MVDR) features for speech recognition have been shown to be better than MFCC features.

The other two features used in this paper are:

- **Log-MEL Features (lMEL)**: Motivated by the physiology of human hearing, the mel scale developed by [18] is applied after performing a short-time Fourier transformation of the audio. In large DNN AMs lMEL features tend to outperform MFCC.

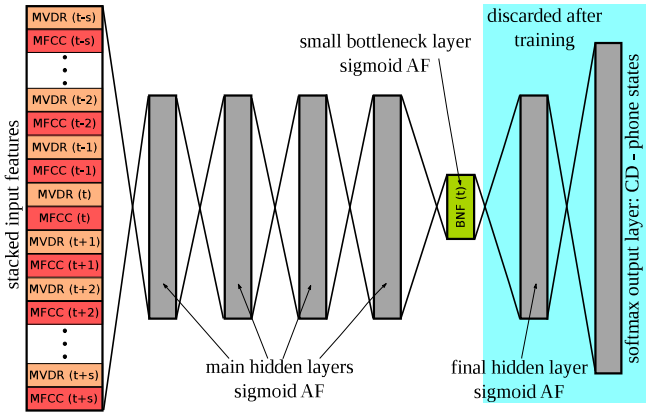

Figure 2: *An example DBNF with 4 hidden layers prior to the bottleneck layer*

- **Tonal Features**: The tonal features used are a combination of Kjell Schubert's [19] pitch based features and Laskowski's [14] FFV features. Tonal features are not suitable as stand-alone features and are only used to augment the other features.

4. Multifeature Modular Deep Neural Network Accoustic Models (mDNN-AMs)

In this section the proposed modular deep neural network acoustic model is introduced. Feed forward DNN AM such as the one depicted in figure 1 have been analyzed and examined by many authors [20, 21]. Both their observations and ours indicate that the addition of further hidden layers does not result in any noticeable improvement for DNNs after about 5-8 layers. One possible explanation is that the lower layers are being poorly trained because the gradient decreases with each layer it is passed back though. In order to solve this problem the mDNN-AMs use a well trained bottleneck feature (BNF) module as the basis for the DNN-AM.

4.1. Multifeature Bottleneck Feature (BNF) Modules

In an MLP the outputs of a given layer can be thought of as an alternative representation of the input feature vector. A small hidden layer in the middle of an MLP will, therefore, provide compact alternative features, with the number of coefficients being controlled by the number of neurons in the hidden layer.

An example of such a network with 4 hidden layers prior to the bottleneck layer is given in Figure 2. Bottleneck features are discriminatingly trained using context dependent subphone states as targets. After training all the layers following the bottleneck are discarded. In a classic GMM system BNFs are typically used to transform the input feature stream into a stream of bottleneck features which are then stacked over a temporal window. An LDA or PCA is then used to reduce the dimension back to the desired input size for the GMM.

Multifeature-BNFs concatenate multiple different features into a single large input vector. Using combinations of the features described in section 3 seven different BNF modules are analyzed in this paper.

4.2. mDNN Topology

An example modular DNN (mDNN) AM using MVDR+MFCC features is shown in figure 3. Two features (MFCC & MVDR) are extracted at each frame and used together with their neighbours in a stack as the inputs to a BNF network.

The final layers of a modular DNN-AM are same as the final layers in a normal DNN-AM. Instead of a normal input layer the modular DNN-AM has a bottleneck layer which consists of stacked BNF frames from an already fine-tuned BNF network. We refer to those final layers as the classification module (or DNN-module) and after integration the BNF network is referred to as the BNF module. If the classification module has an input context of $2r + 1$ (r-BNF frames before and after the current frame) and the BNF module has a input context of $2s + 1$ then the total network requires an input context of $2(r + s) + 1$ frames. For the BNF frame at $t - i$ the input frames from $t - i - s$ to $t - i + s$ have to be stacked and used as the input to the BNF-module. The BNF-module is applied $2r + 1$ times to generate each of the $2r + 1$ BNF frames in the BNF layer.

4.3. Weight Tying

During fine tuning the weights of the BNF-module are tied. Errors can be propagated back past the BNF-layer into all applications. Weight tying allows the modular DNN-AM to continue on using a single BNF-module. Its weights are updated using the average update:

$$\Delta w_j = \sum_{k=1}^{2r+1} \Delta w_j^k \quad (1)$$

such that a single BNF-module learns to produce BNFs that can be used in any part of a stack BNF layer.

4.4. Integration

Although the total computation cost for a single frame is very high, the BNF frames can be cached and reused for the next frame. At frame t the DNN-module of the example mDNN in figure 3 requires the output of the BNF-module for $2r + 1$ different inputs from $t-r$ to $t+r$. For frame $t-1$ it requires outputs from the BNF-module for the inputs from $t - 1 - r$ to $t - 1 + r$. So with the exception of the output of the BNF-module at $t + r$ all of the required outputs for frame t have already been produced and cached.

The BNF-module is simply used to convert a stream (or multiple streams) of input features into a stream of BNF features which are then used as the input stream for the DNN-module. In an offline setting the stream of input feature vectors from an utterance forms a matrix and the BNF-module converts this matrix into a matrix where the columns are BNFs.

Because it is faster to perform a single matrix times matrix operation using a fast BLAS (Basic Linear Algebra Subprograms) library than it is to perform many vector times matrix operations, it makes sense to compute the first hidden layer for all features at the same time. So in an mDNN, if T features are extracted from the audio of an utterance they are first transformed into T activations of the first hidden layer of the BNF-module, then to T activations of the 2nd, 3rd and so on hidden layers and then to T bottleneck features followed by T activations of the first hidden layer of the DNN-module. After transitioning though all the hidden layers the T probability distributions over the cd-phone states are all produced at the same time.

If T features are extracted from the utterance then computing the BNF at frames $t = T$ or $t = 1$ could be problematic since these frames require information about the features at $t = T + r$ or $t = -r$. To solve this every requested feature vector prior to the first one is set to the value of the first feature and the final feature vector is used for every feature that could follow it. The same out of bounds rule is applied when the BNFs are used as an input to the DNN-module of the mDNN.

4.5. Modular DNNs with multiple BNF-modules

The modular DNN is not restricted to a single BNF-module and can use multiple BNF-modules at the same time. Figure 4 shows an example mDNN with two 4 layer BNF-modules. The upper BNF-module uses MFCC features as its input and the lower BNF-module uses MVDR features as its input. Although both networks in this example have an input window of $2s + 1$ frames they could, if it were desired, have different sized input windows and they could also have a differing number of hidden layers. The outputs of both module's BNF layers are concatenated and stacked over a $2r + 1$ window.

5. Experimental Evaluation

All experiments are performed on both the German 2010 Quaero evaluation set (eval2012 [22]) which contains 3 hours and 34 minutes of broadcast news and conversational speech as well as on the 2 hour German IWSLT 2012 development set (dev2012 [23]) that contains TED talks. The results of the systems are measured using WER and reported with an accuracy of two decimal places for the larger eval2012 test set and with an accuracy of a single decimal place for smaller dev2012 test set on which differences smaller than 0.1% would not be statically significant. Statical significance is measured using McNemer's test of significance.

5.1. Speech Recognition System

The decoding and GMM AM training uses the *Janus Recognition Tool-kit* (JRTk) with the Ibis single pass decoder [24].

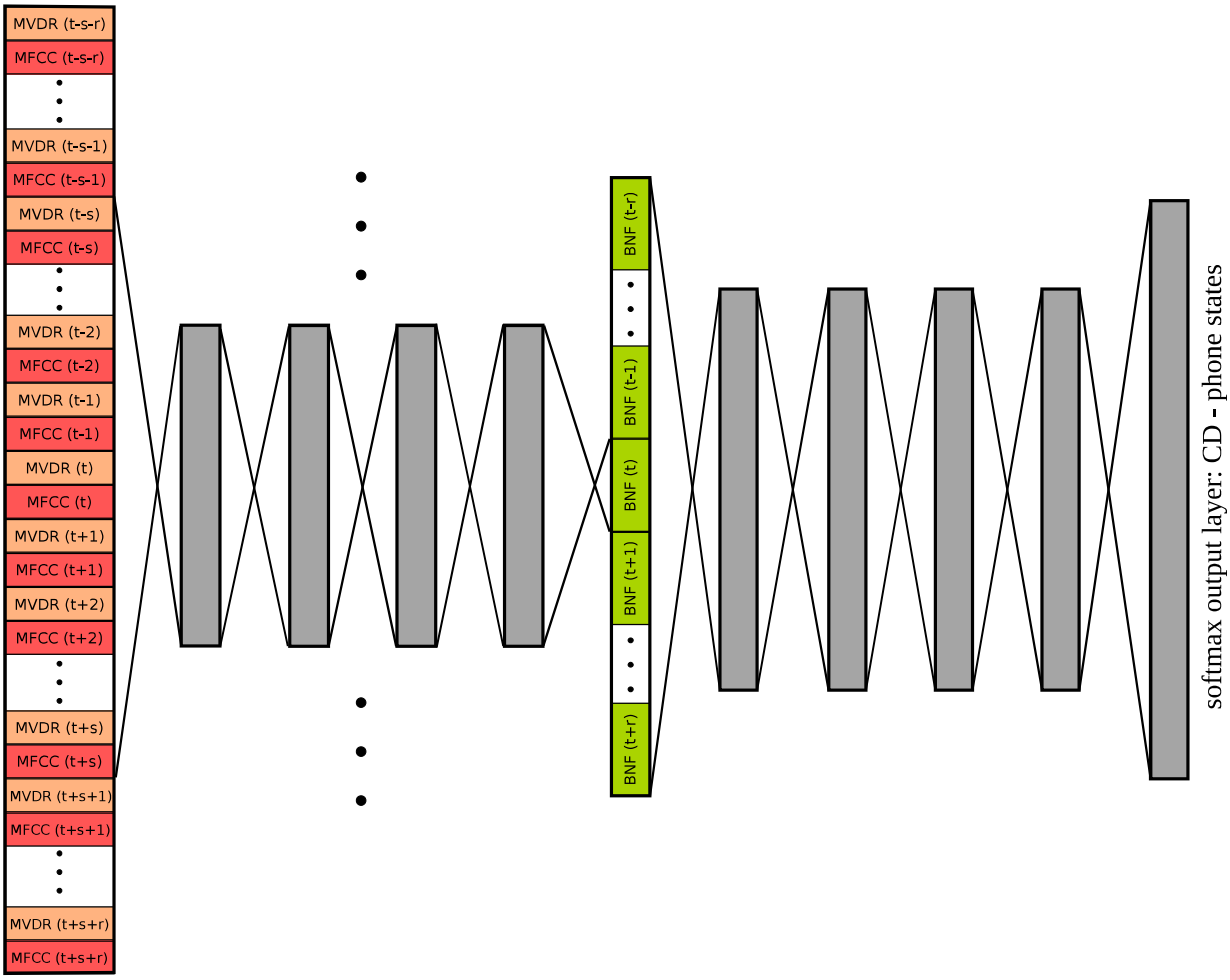

Figure 3: *An example mDNN with 4 hidden layers in the BNF-module and 4 hidden layers in the DNN-module. The input of the DNN-module requires* $2r+1$ *outputs of the BNF-module at different frames. The BNF-module uses a* $2s+1$ *frame window as its input so the whole mDNN network requires* $2(s+r)+1$ *input features which in the example are a concatenation of both MVDR and MFCC features.*

The JrTk [24] is extended with a DNN AM object that implements the same interface to the Ibis decoder as the existing GMM AM. A diverse range of topologies are supported by allowing their computation to be controlled by a tcl script. All acoustic models use a left to right HMM without skip states where each of the 46 normal phonemes have three HMM states and the silence phoneme has only a single state. The cluster tree is built with 6016 leaves.

5.2. Neural Network Training

Each neural network is pretrained layerwise using denoising autoencoders with a 20% corruption and a constant learning rate for 2 million mini batches. After pretraining the final layer is added, with the output layer using the softmax activation function. The full DNN is then fine-tuned using the newbob learning rate schedule. All training is performed using Theano[25].

5.3. Analysis of Multifeature DNNs

Since tonal features can only be used as augmented features and not as individual feature the following combination of input feature were tested:

- Single feature: MFCC, MVDR, lMEL
- 2 features: MFCC+MVDR, lMEL+T[1]
- 3 features: MFCC+MVDR+T
- 4 features: MFCC+MVDR+lMEL+T

Both the MVDR and MFCC features use 20 coefficients while the lMEL features have 40 coefficients and are the same size as the merged MVDR+MFCC feature vector. The addition of 14 tonal features brings the input sizes up to 54 for both the MVDR+MFCC+T (*m2+t*) and the lMEL+T

[1]T=tonal

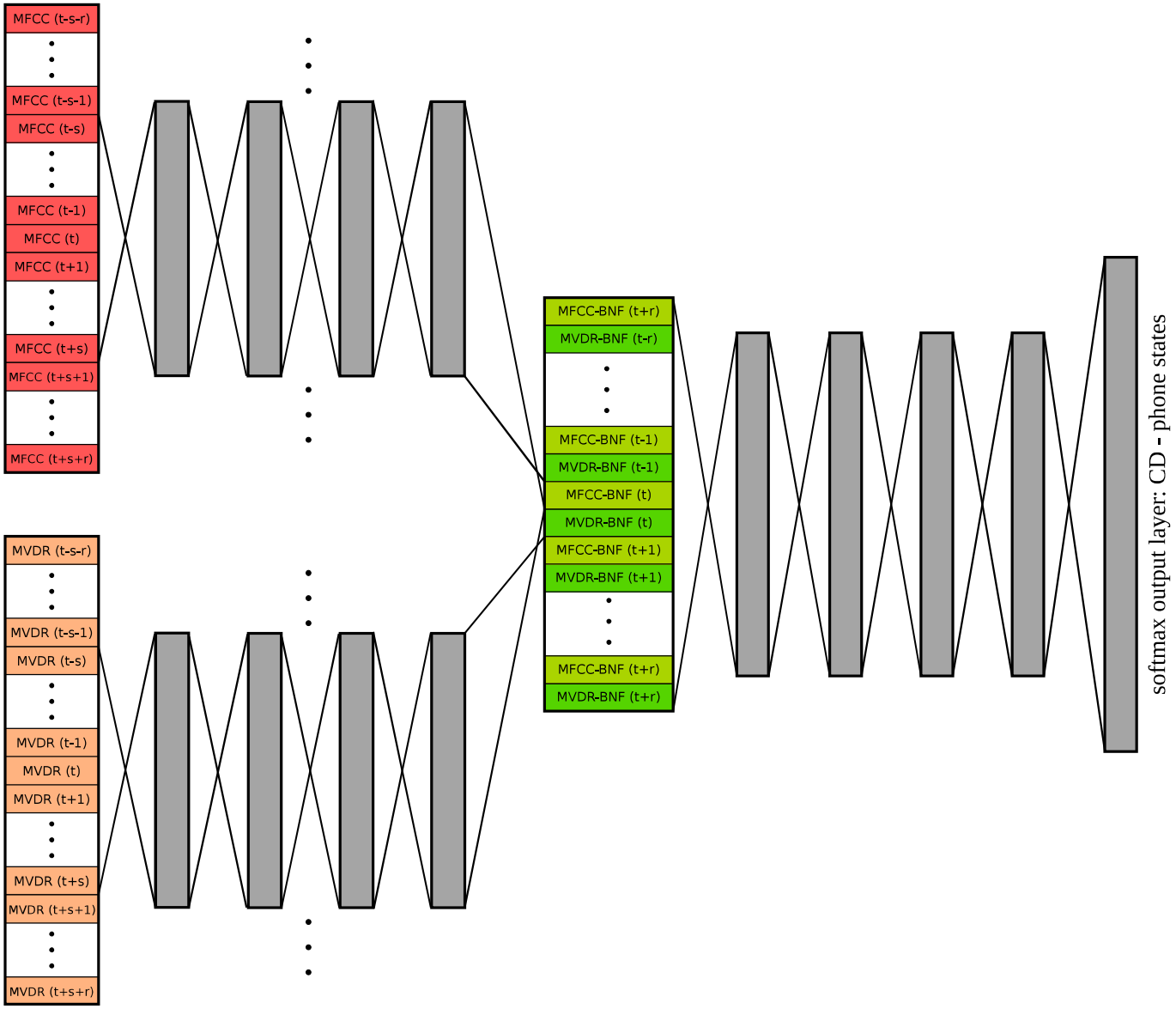

Figure 4: *An example mDNN with a 4 layer DNN-module built on top of two BNF-modules: a 4 layer MFCC BNF-module and a 4 layer MVDR BNF-module. The input of the DNN-module requires* $2r + 1$ *outputs of both the BNF-modules at different frames.*

(*lmel+t*) networks. The final MVDR+MFCC+T+lMEL (*m3+t*) MLP that contains all available features has an input of 94.

Each feature combination is trained on multiple topologies that only vary in their size (1200-2000) and number (4-8) of hidden layers. The reported numbers always use the best topology for the feature combination.

Combining the MVDR and MFCC input features results in a network with a significantly ($p < 0.005$) lower WER than either of the individual features and on about par with networks using the lMEL feature that have the same number of coefficients. The lMEL system is 0.06% better on the eval2010 test set but 0.2% poorer on the dev2012 test set. A pattern in the results shown in table 1 can be found that suggests that input features using more coefficients tend to perform better. The addition of tonal features always results in an improvement.

The best results on the eval2010 test set are achieved by using all input features which is slightly but significantly ($p < 0.005$) better than the *lmel+t* DNNs.

5.4. Evaluation of Multifeature mDNNs

The mDNN topology is evaluated using the same combinations of input features used in the previous experiment and compared with those results. The BNF-modules used in the experiment are taken from working GMM systems where the use of the multifeature deep BNFs were evaluated.

	BNF-Module Name	Features	Best normal DNN		Modular DNN		Comparable CNC	
			eval2010	dev2012	eval2010	dev2012	eval2010	dev2012
MFCC	mfcc	1	15.88	20.3	15.35	19.5	-	-
+MVDR	m2	2	15.45	19.9	14.71	19.4	15.55	20.0
+T	m2+t	3	14.96	19.8	14.54	19.3	-	-
+lMEL	m3+t	4	14.72	19.4	**14.31**	**18.9**	15.09	19.6
lMEL	lmel	1	15.31	20.1	14.72	19.5	-	-
+T	lmel+t	2	14.77	19.6	14.52	19.0	-	-
MVDR	mvdr	1	15.58	20.2	14.81	19.5	-	-

Table 2: *Results of the multifeature mDNNs compared with both normal DNN using the same input feature combinations and equivalent confusion network combinations. Tested on both the eval2010 and dev2012 test set.*

	eval2010	dev2012
MFCC	15.88	20.3
+MVDR	15.45	19.9
+T	14.96	19.8
+lMEL	**14.72**	**19.4**
lMEL	15.31	20.1
+T	14.77	19.6
MVDR	15.56	20.2

Table 1: *Evaluation of DNNs using various combinations of MFCC, MVDR, T and lMEL input features. Result presented on the IWSLT dev2012 and Quaero eval2010 test sets.*

The DNN-modules are pretrained by first mapping the input features into the bottleneck feature space and performed by training and stacking denoising autoencoders. After pretraining the classification layer is added and the whole mDNN network is jointly finetuned. The input feature sizes range from 20 for the mfcc network features to 94 for the m3+t network. With r, the number of BNF frames before and after the current frame used as the input to the DNN-module, set to 7 and each BNF layer containing 42 neurons the DNN-modules input layer has 630 neuron. All mDNN networks have the same topology. Both their BNF-modules and their DNN-modules have 4 hidden layers of 2000 neurons. The whole network, therefore, has 9 hidden layers.

The results of this experiment are shown in table 2. As a comparison, for each input feature combination its best result with a normal DNN, regardless of the topology, is shown in columns of the table. The last column contains a comparison to a system combination using confusion networks performed on the DNN output lattices of single feature DNNs. The cnc comparison result in line two of table is a combination of the best MVDR DNN and the best MFCC DNN and although it is slightly better than both of them it is outperformed by both the MVDR+MFCC DNN and the MVDR+MFCC mDNN. The cnc comparable to the m3+t network is a combination of the MFCC DNN, the MVDR DNN, and the *lmel+t* DNN and does not even improve on the performance of the lmel+t DNN.

For all input feature combinations the mDNN outperforms the normal DNN by 0.5% absolute or more on the dev2012 test set. On the eval2010 test set the improvements varied from an improvement of 0.25% on the lmel+t features to over 0.7% on both the MVDR and MVDR+MFCC features. The relative usefulness of features is not altered by using an mDNN. With 19.4% on dev2012 the *m3+t* DNN has 4.5% relative lower WER than the basic MFCC DNN which has a WER of 20.3 and a 3.5% lower WER than the lMEL DNN which is the best single feature DNN. In the modular case the improvements are slightly less. All single feature mDNNs have a WER on dev2012 of 19.5% and the *m3+t* network has a 3% lower WER at 18.9%. For the single feature inputs the mDNN results in improvements of 3-4% compared to the normal DNN while the multifeature inputs are only improved by 2.5-3.5%.

Using only lMEL features as inputs performs as well as using the combined MVDR+MFCC feature in both the DNN and the mDNN and on both test sets. The addition of tonal features boosts the performance of the lMEL DNN and mDNN more than the MVDR+MFCC DNN and mDNN.

In total the best multifeature mDNN reduces the WER of a basic MFCC DNN by 7% relative from 20.3% to 18.9% on the dev2012 test set and by 10% relative from 15.88% to 14.31% on the eval2010 test set. Compared to the best single feature DNN, lMEL, it still improves the dev2012 test set by 6% and the eval2010 test set by 6.5%.

5.5. Evaluation of mDNNs with multiple BNF-modules

The effectiveness of the mDNN with multiple BNF-modules is evaluated by training 8 mDNNs with between two and seven BNF-modules. The results are compared to performing a CNC on normal DNN networks that use the same input features and the BNF-modules. The BNF-modules are the same as in the multifeature mDNN experiment and can themselves contain multiple input features. After mapping the training data into the bottleneck feature spaces of all BNF-modules used. The DNN-module is pretrained on the merged BNF features. All other training parameters are the same as in the previous experiments.

In all cases the mDNN outperformed the confusion network combination of DNN systems using the same input features. The best mDNN with multiple BNF-modules m2+t $\oplus$ lmel+t $\oplus$ m3+t ($\oplus$ is used to indicate that a combination of

	BNF-Modules	Features	Name	Modular DNN		Comparable CNC	
				eval2010	dev2012	eval2010	dev2012
mfcc	1	1		15.35	19.5	-	-
⊕ mvdr	2	2	sys01	14.54	19.2	15.55	20.0
⊕ m2	3	2	sys03	14.73	19.3	15.44	19.9
mfcc ⊕ mvdr ⊕ lmel+t	3	4	sys02	14.24	18.7	15.09	19.6
m2+t ⊕ lmel+t	2	4	sys04	14.19	18.8	14.68	19.4
⊕ m3+t	3	4	sys08	14.06	18.7	14.45	19.2
⊕ mfcc ⊕ mvdr	5	4	sys06	14.33	18.9	14.83	19.4
⊕ m2 ⊕ lmel	7	4	sys05	14.44	18.8	14.69	19.4
m2 ⊕ lmel	2	4	sys07	14.34	19.1	15.07	19.8

Table 3: *Comparison of mDNNs using multiple BNF-modules with confusion network combinations of normal DNNs using the same input features. The ⊕ is used to indicate that multiple BNF-modules are combined in a single mDNN.*

BNF-modules) improves the best single module mDNN by 0.2% from a WER 18.9% to 18.7% on the dev2012 test set and by 0.25% from 4.31% to 14.06% on the eval2010 test set. Using McNemar's significance test this is found to be significant at $p < 0.005$. The overview of the results given in table 3 begins with a single BNF-module mDNN using mfcc input features that achieves a WER of 15.35% on eval2010 and 19.5% on dev2012. The next entry augments that mDNN with an MVDR BNF-module and improves dev2012 by 0.3% to 19.2% and eval by 0.81% from 15.35% to 14.54%. The further addition of the MVDR+MFCC BNF-module degraded the dev2012 test set to 19.3% and the eval2010 test set to 14.73%. If instead of the MVDR+MFCC BNF-module the lmel+t BNF-module is added to the mfcc ⊕ mvdr mDNN then it is further improved to 14.24% on eval2010 and 18.7% on dev2012.

The best mDNN with two BNF-modules is the m2+t ⊕ lmel+t mDNN which has a WER of 14.19% on eval2010 and 18.8% on dev2012. The addition of an m3+t BNF-module improves it slightly by 0.13% to 14.06% on the eval2010 test set and by 0.1% to 18.7% on the dev2012. The further inclusion of both the MVDR BNF-module and the MFCC BNF-module slightly increases the WER on both sets. Increasing the number of BNF-modules to 7 by also including the m2 and lmel BNF-modules into the mDNN results in another slight increase in WER.

The usefulness of tonal features can be clearly seen by comparing the m2 ⊕ lmel mDNN to the m2+t ⊕ lmel+t DNN which add tonal features to the input to both of the BNF-modules. They are able to improve the dev2012 test set by 0.3% and the eval2010 test set by 0.15%.

Using an mDNN with multiple BNF-modules increases the mDNNs overall improvement compared to an MFCC-DNN by 8% relative on the dev2012 test and by 11.5% on the eval2010 test set. Compared to an lMEL DNN it reduced the WER by 7% relative from 20.1% to 18.7% on dev2012 and by 8% relative from 15.31% to 14.07%.

6. Conclusion

The modular deep neural network acoustic model presented in this paper incorporates the well trained feature extraction networks using multiple input features. It is initially evaluated using only a single feature extraction module. This evaluation demonstrates the usefulness of using multiple different input feature vectors. Modular deep neural networks, whose sole feature extraction network uses multiple features, outperform those using fewer features or even a single feature.

Using two or more different feature extraction networks as modules in the same modular deep neural network results in further improvements. The best approaches use three feature extraction networks that are, in turn, each trained using multiple input features. The best modular deep neural network is able to reduce the word error rate on the test data sets by up to 11.5% relative improvement compared to a baseline deep neural network..

7. Acknowledgements

The authers would like to thank the reviewers for their constructive comments.

The work leading to these results has received funding from the European Union under grant agreement $n \circ 287658$.

8. References

[1] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech & Language*, vol. 14, no. 4, pp. 373–400, 2000.

[2] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*. IEEE, 1997, pp. 347–354.

[3] J. Gehring, W. Lee, K. Kilgour, I. R. Lane, Y. Miao, A. Waibel, and S. V. Campus, "Modular combination

of deep neural networks for acoustic modeling." in *INTERSPEECH*, 2013, pp. 94–98.

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

[5] H. SAWAI, Y. MINAMI, M. MIYATAKE, A. WAIBEL, and K. SHIKANO, "Connectionist approaches to large vocabulary continuous speech recognition," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 74, no. 7, pp. 1834–1844, 1991.

[6] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, p. 310, 1995.

[7] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8614–8618.

[8] D. Yu and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural networks." in *INTERSPEECH*, vol. 237, 2011, p. 240.

[9] C. Plahl, R. Schlüter, and H. Ney, "Improved acoustic feature combination for lvcsr by neural networks." in *INTERSPEECH*, 2011, pp. 1237–1240.

[10] C. Plahl, M. Kozielski, R. Schluter, and H. Ney, "Feature combination and stacking of recurrent and non-recurrent neural networks for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6714–6718.

[11] K. Kilgour, T. Seytzer, Q. Nguyen, and A. Waibel, "Warped minimum variance distortionless response based bottle-neck features for LVCSR," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[12] The National Institute of Standards and Technology, "NIST Open Keyword Search 2013 Evaluation (OpenKWS13)," http://www.nist.gov/itl/iad/mig/openkws13.cfm, Apr. 2013, last accessed: July 3, 2013.

[13] F. Metze, Z. A. Sheikh, A. Waibel, J. Gehring, K. Kilgour, Q. B. Nguyen, and V. H. Nguyen, "Models of tone for tonal and non-tonal languages," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 261–266.

[14] K. Laskowski, M. Heldner, and J. Edlund, "The fundamental frequency variation spectrum," *Proceedings of FONETIK 2008*, pp. 29–32, 2008.

[15] J. Gehring, Q. B. Nguyen, F. Metze, and A. Waibel, "Dnn acoustic modeling with modular multi-lingual feature extraction networks," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 344–349.

[16] M. Wölfel, J. W. McDonough, and A. Waibel, "Minimum variance distortionless response on a warped frequency scale." in *INTERSPEECH*, 2003.

[17] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[18] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[19] K. Schubert, "Pitch tracking and his application on speech recognition," Ph.D. dissertation, Diploma Thesis at University of Kalsruhe (TH), Germany, 1998.

[20] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14–22, 2012.

[21] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[22] S. Stüker, K. Kilgour, and F. Kraft, "Quaero speech-to-text evaluation systems," in *High Performance Computing in Science and Engineering '11*. Springer Berlin Heidelberg, 2012, pp. 607–618.

[23] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and M. Federico, "Report on the 10th iwslt evaluation campaign," in *Proceedings of the International Workshop on Spoken Language Translation, Heidelberg, Germany*, 2013.

[24] H. Soltau, F. Metze, C. Fugen, and A. Waibel, "A one-pass decoder based on polymorphic linguistic context assignment," in *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*. IEEE, 2001, pp. 214–217.

[25] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010, oral Presentation.