# Convolutive Bottleneck Network Features for LVCSR

Karel Veselý [1], Martin Karafiát [2], František Grézl [3]

*Speech@FIT, Brno University of Technology*
*Božetěchova 2, 612 66 Brno, Czech Republic*
[1] iveselyk@fit.vutbr.cz
[2] karafiat@fit.vutbr.cz
[3] grezl@fit.vutbr.cz

***Abstract*—In this paper, we focus on improvements of the bottleneck ANN in a Tandem LVCSR system. First, the influence of training set size and the ANN size is evaluated. Second, a very positive effect of linear bottleneck is shown. Finally a Convolutive Bottleneck Network is proposed as extension of the current state-of-the-art Universal Context Network. The proposed training method leads to 5.5% relative reduction of WER, compared to the Universal Context ANN baseline. The relative improvement compared to the 5-layer single-bottleneck network is 17.7%.**

**The dataset *ctstrain07* composed of more than 2000 hours of English Conversational Telephone Speech was used for the experiments. The TNet toolkit with CUDA GPGPU implementation was used for fast training.**

## I. Introduction

In past years, significant research interest was devoted to the applications of Artificial Neural Networks (ANN) in Automatic Speech Recognition (ASR) systems.

The idea of Hybrid ASR was first formulated by Bourlard in the pioneering work [1], where the ANN produces phoneme posteriors, which are log-transformed and used as emission probabilities of HMM states. This approach is known as Pure Hybrid ASR.

Another popular class of Hybrid ASR systems are Tandem systems [2]. Here, the ANN plays role of discriminative feature extractor for subsequent GMM-HMM model. In this case, the ANN is trained to produce vectors of phoneme or phoneme-state posteriors. Often, the posteriors are post-processed by PCA. The Tandem system is more complex than the Pure Hybrid system. The good point is that with Tandem system, we can easily apply all known GMM-HMM tricks such as discriminative training and speaker adaptive training.

As described in [3], further accuracy improvement as well as system simplification (no PCA) can by achieved by using Tandem with Bottleneck features (BN-features). In this case, the features are obtained directly from a hidden layer with low dimensionality. Again, the ANN is trained to classify phoneme states, therefore we assume that the BN-features contain concentrated discriminative information.

The state-of-the-art BN-feature systems are based on Universal Context [4]. Conceptually, it is a tandem of two bottleneck ANN classifiers, where the first ANN performs approximative classification from a short time-context, while the second ANN performs more precise decision based on longer temporal context.

In this paper, we focus on improving the BN-feature extractor. We train on the *ctstrain07* corpus composed of more than 2000 hours of 8KHz Conversational Telephone Speech (CTS). We can afford to train on such huge dataset thanks to the *TNet toolkit* [5] which contains fast implementation of stochastic gradient descent benefitting from CUDA GPGPU.

All the BN-feature optimizations are proposed in section II. First, the dataset size and model size are tuned (section II-A). Then, the Linear bottleneck is proposed (II-B), which is followed by brief discussion of Universal Context ANN (II-C). Finally, the Convolutive Bottleneck Network is proposed (II-D). The rest of the paper contains the experimental setup description (section III), results (section IV) and conclusion (section V).

## II. Bottleneck Features

### A. Scaling the Single Bottleneck System

The performance of any ANN depends on two principal factors: 1) the amount of training data and 2) the number of trainable parameters. Theoretically, the perfect model would have infinite number of trainable parameters, that would be precisely estimated on infinite amount of training data. However, this is not practically feasible because the training algorithm would have infinite running time. Based on the empirical experience, the training time is nearly linearly dependent on both the set-size and model-size while keeping the other size fixed.

The same applies for the bottleneck ANNs, only the topology is different. Typical Bottleneck ANN is composed of 5 layers, where the middle layer has only a few tens of neurons. The activations of these neurons are used as features. First, several different amounts of training data were used to train fixed topology ANN. Then, several different sized networks were trained on 1000 hours.

According to the machine learning theory of the frequentist models [6], we might expect following behavior: By adding more training data with fixed ANN topology, the performance improves until the model parameters are reliably estimated. Adding further data is a waste of time. However, it is still possible to improve the performance by adding more trainable parameters, which consequently increases the set-size needed for reliable estimation of the parameters. Unfortunately, this joint enlargement of the ANN training causes quadratic growth of the training time.

## B. Linear Bottleneck

In the pioneering work on bottleneck features (Grézl et al. [3]) 5-layer bottleneck ANNs with logistic sigmoid units were used in all the 3 hidden layers including the bottleneck. It was reported that per-frame classification accuracy (measured on a cross-validation set) of such 5-layer bottleneck ANN drops by 3% absolute compared to a 4-layer ANN without bottleneck. In other words, the presence of a bottleneck in the ANN deteriorates the amount of discriminative information that is propagated through the ANN towards the posteriors. The respective performance loss was found to be dependent on the bottleneck size.

Interestingly, we will show that part of this deterioration can be compensated by using linear units in the bottleneck. Our typical bottleneck size is 30 units, which represents an aggressive compression. The hypothesis is that in case of low bottleneck dimensions, the linear units allow to encode more discriminative information than the sigmoid units. The previous statement is empirically supported by the observation of lower absolute covariances in normalized features for the case of linear bottleneck.

Now we will focus on what the substitution of a sigmoid bottleneck by a linear bottleneck means in theory. The propagation through a sigmoid bottleneck can be expressed as:

$$\mathbf{a_3} = \mathbf{W_3}\,\sigma\left(\mathbf{W_2}\mathbf{h_1} + \mathbf{b_2}\right) + \mathbf{b_3} \qquad (1)$$

where $\sigma$ is logistic sigmoid; $\mathbf{a_3}$ is vector of activations of the 3rd hidden layer; $\mathbf{h_1}$ is vector of 1st hidden layer output; $\mathbf{b_i}$ is $i$-th layer bias vector and $\mathbf{W_i}$ is $i$-th layer weight matrix. The whole situation is shown in Fig. 1.

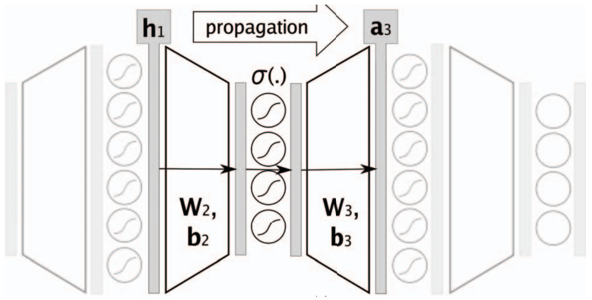

Figure 1. Illustration of bottleneck in 5-layer ANN

By using linear bottleneck, the propagation (1) simplifies to:

$$\mathbf{a_3} = \mathbf{W_3}\left[\mathbf{W_2}\mathbf{h_1} + \mathbf{b_2}\right] + \mathbf{b_3}\ , \qquad (2)$$

by simple rearranging we get:

$$\mathbf{a_3} = \left[\mathbf{W_3}\mathbf{W_2}\right]\mathbf{h_1} + \left[\mathbf{W_3}\mathbf{b_2} + \mathbf{b_3}\right]\ , \qquad (3)$$

where we see that the propagation through the two bottleneck-neighbouring layers can be expressed as a propagation through a single layer, where the weight matrix has limited rank.

## C. Universal Context Network

Another possibility to improve the BN-features is to experiment with different forms of hierarchical ANN structures. Recently, inspired by the idea of Split Time Context introduced by Schwarz [7], the bottleneck networks were extended to the Universal Context Networks [4].

In the *Split Time Context* approach, the network hierarchy has two levels. On the first level, different parts of temporal trajectories of input features are modeled by separate ANNs. The second level consists of a merger ANN which fuses the posteriors from the first level. This hierarchical structure was found to be beneficial especially in case of limited training data such as for the TIMIT database.

In the *Universal Context* approach, the ANN is also hierarchical. The primary network is a 5-layer Bottleneck ANN, that could be used as feature extractor in Tandem system. However the key point is that part of the primary ANN is used as feature extractor for the secondary 5-layer Bottleneck ANN. Time window with time-domain sub-sampling is used to select outputs from the primary ANN to form the input of the secondary one. The training of the Universal Context ANN is done in two phases:

*1) First Phase:* primary 5-layer Bottleneck ANN is trained on short time context of 11 frames. The ANN is trimmed to have the activations of the bottleneck as output.

*2) Intermezzo:* activations of the bottleneck are mean- and variance-normalized. Context expansion by concatenation of frames with time offsets -10 -5 0 5 10 is performed. The overall time context is now 31 frames.

*3) Second Phase:* the secondary 5-layer Bottleneck ANN is trained, while the parameters in the torso-ANN from the first phase are fixed. Finally the secondary ANN is trimmed in order to produce the activations in the bottleneck, which are already the final features for GMMs.

The two training steps are shown in Fig. 2, the primary ANN is on the left (I.), the trimmed parameter-locked ANN and the secondary ANN are on the right side (II.). The context expansion CTX between the two networks is also marked.

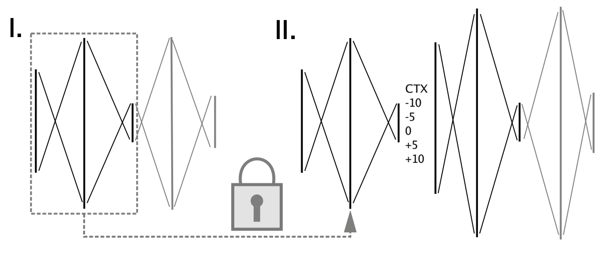

Figure 2. Two phases of the Universal Context network training

The *Second phase* of the *Universal Context ANN* training is not ideal. The fact that some model parameters are fixed while some are trained is clearly not optimal.

### D. Convolutive Bottleneck Network

Although it might seem difficult to backpropagate through the context expansion, the solution is possible. As can be seen in Fig. 3, the problem can be overcome by putting the context expansion from between the two networks to the front of the first ANN, while the first ie. the torso-ANN will be cloned to 5 instances with shared weights.

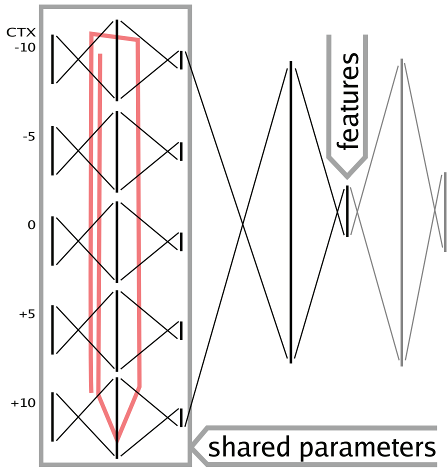

Figure 3. Convolutive Bottleneck Network (CBN) structure

This form of ANN topology can be considered as a Convolutional network. It complies with all the three attributes mentioned by Bishop [6]: 1) *local receptive fields* – by using short time context as input for each torso-ANN in globally longer context, 2) *shared parameters* and 3) *subsampling* – by context expansion with a time-step of 5 frames. Therefore, such form of ANN topology will be called *Convolutive Bottleneck Network* (CBN).

The CBN can be trained directly from random initialization. However taking into account that the CBN is a 7-layer *deep architecture*, we might expect *gradient vanishing effect* [8], therefore some form of pre-training is advisable. Analogically with the Universal Context case, the first two layers will be pre-trained as a trimmed part of 5-layer Bottleneck ANN.

Three training strategies are applicable:

*1-Pass:* The CBN is trained directly from random initialization.

*2-Pass:* First, the torso-ANN is pre-trained as in the Universal Context case, then CBN is built and all the parameters are trained.

*3-Pass:* The torso-ANN is pre-trained, then the CBN network is built. One iteration is performed while keeping the shared part fixed. In the third pass, all the parameters are trained.

All the three proposed strategies will be evaluated in the experimental part.

Another important trick in the training is that the updates of shared parameters should be scaled down by the inverse number of sharing of such parameters. In our case we scale by $\frac{1}{5}$.

The ANNs of both the CBN form as well as the Universal Context form contain two bottlenecks. These can be composed of either linear or sigmoidal units. The best combination will be decided in the experimental part.

## III. Experimental Setup

*Database:* The initial GMM models were trained on the *ctstrain04* training set which is a subset of h5train03 training set defined at Cambridge University. It consists of Switchboard1, Switchboard2 and Call Home English data. Sentences containing words, which do not occur in the training dictionary were removed. The total amount of training data was 278 hours.

The ctstrain04 set was further extended by data from Fisher 1 and 2 corpora. The resulting *ctstrain07* data set comprises 2000 hours of data. For the ANN training, we were randomly selecting utterances from the ctstrain07 dataset in order to reach the desired set-size. The disjoint cross-validation set was also selected from the ctstrain07 set and was fixed to 100 hours.

All the Tandem systems were tested on the *Hub5 Eval01* test set. It is composed of 3 subsets of 20 conversations from Switchboard-1, Switchboard-2 and Switchboard-cellular corpora, for a total length of more than 6 hours of audio data.

*Initial acoustic models:* The speech recognition system is based on HMM cross-word tied-states triphones. The initial acoustic models were trained from scratch using mixture-up training on ctstrain04 set. The resulting models contained $\approx$8500 tied states and 24 Gaussian mixtures per state. Finally, Heteroscedastic Linear Discriminant Analysis (HLDA) transform was estimated and models were retrained in the new space. The PLP features with 13 coefficients and applied Vocal Tract Length Normalization (VTLN) were expanded with derivatives $\Delta$, $\Delta^2$ and $\Delta^3$ and transformed by HLDA to 39 dimensions. The system with PLP-HLDA features was used to generate forced alignments for ANN training. There were 120 labels/target classes corresponding to HMM states of 40 English phonemes including silence. Each phoneme is modelled by 3-states.

*ANN Parameterization:* Long temporal context parameterization as proposed in [7] was used. The parameters are 15 log Mel-filterbank outputs derived with 25ms window, 10ms shift and applied VTLN. The parameters were per-speaker mean- and variance-normalized. In each band, a temporal context is taken, scaled by Hamming window and compressed by Discrete Cosine Transform (DCT). In case of simple 5-layer Bottleneck ANNs, the temporal context of 31 frames was used with 16 basis DCT (including C0). For Convolutive Bottleneck Network or Universal Context Network, the 11-frame temporal context was used with 6 basis DCT (including C0). By concatenating DCT coefficients for all 15 bands, we obtain feature vectors of 240 or 90 coefficients. Such network inputs were finally globally mean- and variance-normalized.

*ANN Topologies:* The feature-producing bottleneck size is always 30. The dimensionality of the input is given by the used parameterization (240 or 90) and the dimensionality of the output is always 120 (number of phoneme states). The remaining "free" hidden layer sizes are constrained to be equal and are calculated to fit the desired number of parameters.

In case of the Convolutive Bottleneck Networks or the Universal context network, the output of the torso-ANN has always 80 dimensions.

Three activation functions were used: In hidden layers the neurons were by default sigmoidal, for some hidden bottlenecks the neurons were linear. In case of output layer softmax was used.

*ANN Initialization:* The weight matrices were initialized by Normal distribution scaled by 0.1, the biases of sigmoid units were initialized uniformly from interval -4.1,-3.9 and the biases of the linear units were set to zero.

*ANN Training:* Stochastic gradient descent optimizing cross-entropy loss function was used. The learning rate was scheduled by the "newbob" algorithm: *"The learning rate is kept fixed as long as the single epoch increment in cross-validation frame accuracy is higher than 0.5%. For the subsequent epochs, the learning rate is being halved till the cross-validation increment is inferior to stopping threshold 0.1%."* The ANN weight updates were performed per blocks of 512 frames with various initial learning rates depending on the ANN architecture.

*Features for ASR:* The BN-features produced by different ANNs were transformed by Maximum Likelihood Linear Transform (MLLT), which considers HMM states as classes. The transformed bottleneck features were used either raw, expanded by derivative $\Delta$ or concatenated with PLP-HLDA features. The final features were mean- and variance-normalized.

New models were trained by single pass retraining from the PLP-HLDA *initial acoustic models*. Next, 18 maximum likelihood iterations followed to better settle new HMMs in the new feature space.

The test set was decoded with bigram language model taken from AMI speech recognition system for NIST Rich Transcriptions 2007 [9], while the CMU dictionary was used.

## IV. Results

### A. Scaling Single Bottleneck System

In the first set of experiments, we evaluated the influence of adding more training data, with fixed ANN topology. In Fig. 4, we see that adding more data always improves cross-validation frame accuracy (left y-axis). However, the improvement by adding data beyond 1000 hours is small. On the right y-axis, we see that the epoch duration grows linearly with the set-size. The topology was 5-layer ANN with 1 million parameters. The 1000 hours subset had optimal performance/time ratio and is used for further experiments.

In the second round of experiments, we optimized the model size. As can be seen in Tab. I, by adding several million of parameters (1st column), the cross-validation frame accuracy (cvAcc) always improves. However the Word Error Rate

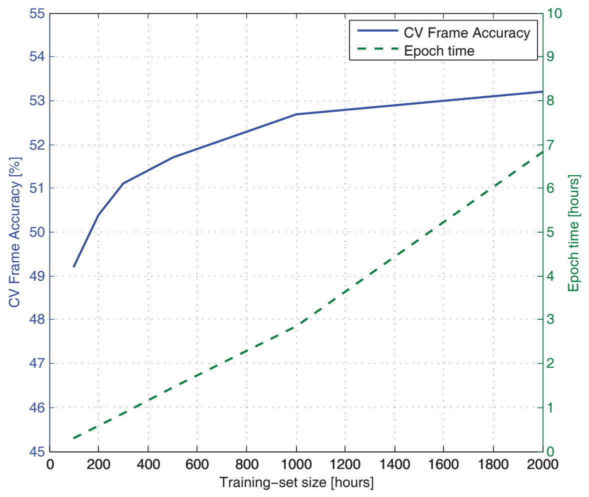

Figure 4. Training 5-layer BN ANN with different set-sizes.

(WER) stops decreasing when using more than 3 million parameters, therefore *3 million parameters will be our preferred model-size*.

Table I
5-Layer BN ANN with Different Parameter Counts, 1000 hours

| Size | Dim | cvAcc [%] | WER [%] | time/iter |
|---|---|---|---|---|
| 1M | 2381 | 52.5 | 33.9 | 2h52min |
| 2M | 4762 | 53.6 | 33.7 | 4h22min |
| 3M | 7143 | 54.2 | 32.8 | 6h13min |
| 4M | 9524 | 55.0 | 32.7 | 7h58min |
| 5M | 11905 | 55.0 | – | 9h46min |
| 6M | 14286 | 55.3 | – | 11h44min |

The topology pattern is *240,Dim,30,Dim,120*, where Dim is second column of Tab. I.

### B. Linear Bottleneck

In the following experiment, the *Sigmoidal bottleneck* was replaced by *Linear bottleneck*. In Tab. II, we see that 1M ANN with Linear-BN has slightly lower WER than much larger 3M Sigmoid-BN ANN. Moreover, the 3M Linear-BN ANN performs better than all the previous networks. From the results, we clearly see that the linear bottleneck is beneficial and leads to WER reduction.

Table II
Linear vs. Sigmoidal Bottleneck, 1000 hours

| Size | linear BN | | sigmoidal BN | |
|---|---|---|---|---|
| | cvAcc [%] | WER [%] | cvAcc [%] | WER [%] |
| 1M | 53.8 | 32.7 | 52.5 | 33.9 |
| 3M | 56.2 | **32.1** | 54.2 | 32.8 |

### C. Convolutive Bottleneck Networks

After all the experiments with 5-layer single-bottleneck ANNs, we started to experiment with 7-layer Convolutional Bottleneck Networks (CBN).

*Bottleneck types:* In the CBN there, are two bottlenecks which can be either Sigmoidal or Linear. Several tens of ANNs were trained in order to decide which combination is the best. The networks were trained directly from random initialization, while a reduced dataset of 100 hours was used. As can

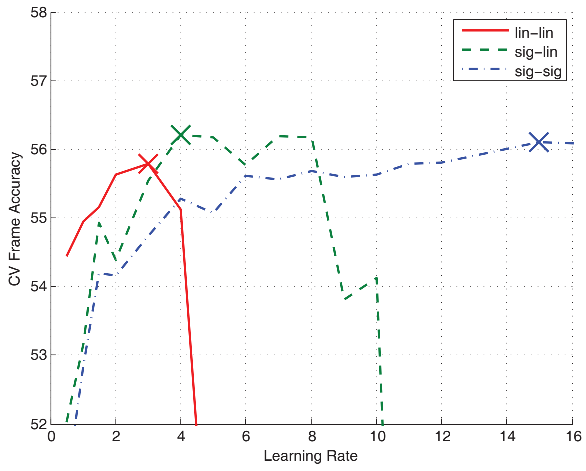

Figure 5. Final Cross-Validation Frame Accuracies as functions of initial learning rate.

be seen in Fig. 5 the CBNs with Linear bottlenecks are particularly sensitive to the choice of the initial learning rate. The highest cross-validation frame accuracy was achieved with a combination of Sigmoidal and Linear bottlenecks for shared-part output and feature-producing bottleneck respectively, for initial learning rate[1] of 4. The maxima from all the three curves were taken and WER was evaluated. The results are in Tab. III,

Table III
DIFFERENT BOTTLENECK-TYPE COMBINATIONS

| Bottlenecks | Learning rate | cvAcc [%] | WER [%] |
|---|---|---|---|
| SigSig | 15 | 56.0 | 31.3 |
| SigLin | 4 | 56.2 | 30.7 |
| LinLin | 3 | 55.8 | **30.5** |

where we see contradictory results. Although the combination of sigmoidal and linear bottlenecks (SigLin) had the best Cross-Validation Frame Accuracy, the best WER was achieved in case of the two linear bottlenecks (LinLin), which had contrarily the worst Cross-Validation Frame Accuracy. This paradox would deserve more analysis. The LinLin model was used for further experiments.

*Multi-pass training:* In the next set of experiments, we have evaluated the effect of pre-training. All three strategies proposed in Sec. II-D were evaluated on the CBN LinLin and 100h dataset. As can be seen in Tab. IV, the lowest WER was achieved with the 3-Pass strategy with the absolute improvement of 0.8% when compared to the 1-Pass baseline.

[1]In the stochastic gradient descent implementation in TNet, the gradient is divided by blocksize as default setting, this is the reason why the learning rate values are higher than usually.

Table IV
MULTI-PASS TRAINING OF CBN (LINLIN), 100 HOURS

| Strategy | cvAcc [%] | WER [%] |
|---|---|---|
| 1-Pass | 55.8 | 30.5 |
| 2-Pass | 55.6 | 30.0 |
| 3-Pass | 55.4 | **29.7** |

Again, we observe the same contradiction, that the best WER is achieved in case of the worst Cross-Validation Frame Accuracy.

## D. Final Evaluation

In the final set of experiments, we wanted to evaluate all the modifications either to the ANN structure or to the training procedure.

Three Tandem systems were trained by Maximum Likelihood for each ANN. The first system used plain BN-features, the second used BN-features extended by time derivatives $\Delta$ and the third one used concatenation of plain BN-features with PLP-HLDA features.

The systems were evaluated on the *eval01* test set. The results in Tab. V are divided into four vertical blocks:

The first block represents the standard 5-layer single-bottleneck ANN with 3 million parameters trained on 1000 hours dataset, this will be the baseline.

In the second block, there are three Universal Context ANNs, where first, a smaller network with 1.5 million parameters was trained on 100 hours dataset. Here we see that despite of training smaller network on smaller dataset, the Universal Context ANN always outperforms the baseline by more than 1.5% absolute. In the next row, the dataset-size was extended to the ideal size of 1000 hours, then the model-size was extended to the optimal 3 million parameters.

In the third block, the two sigmoidal bottlenecks in the UC system were replaced by the two linear bottlenecks. Here we see further improvement of 1% absolute for plain BN-features.

In the fourth block, the UC network was expanded to the Convolutional Bottleneck Network, where on the first line the ANN was trained directly from the random initialization. Finally the effect of pre-training of the shared torso-ANN is shown in the last line. Here, we see that in case of CBN the pre-training is important and leads to absolute improvement of 0.6% in case of the plain BN-features.

In the last row of the Tab. V we see that the BN-features features are no longer complementary with the PLP-HLDA and the WER improvement from the feature fusion is only 0.1% which is not significant.

Finally it should be stated that the WER of the alignment-producing baseline GMM-HMM system was 37.2%.

## V. CONCLUSION

In this paper, we presented advanced techniques to bottleneck feature optimization. We have shown that for large datasets, it is feasible to fully train 3 million ANN parameters

Table V
FINAL TANDEM SYSTEM EVALUATION ON THE EVAL01 TEST SET

| **Type** | **Bottle-necks** | **Size** | **Data** | **Phases** | **cvAcc [%]** | **eval01 WER [%]** | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | **NN** | **NN+Δ** | **NN+PLP** |
| 5L | Sig | 3M | 1000h | - | 54.3 | 32.8 | 32.5 | 31.1 |
| UC | SigSig | 1.5M | 100h | 2 | 53.8 | 31.3 | 30.2 | 29.4 |
| UC | SigSig | 1.5M | 1000h | 2 | 57.0 | 29.7 | 29.1 | 28.5 |
| UC | SigSig | 3M | 1000h | 2 | 58.5 | 28.6 | 27.9 | 27.9 |
| UC | LinLin | 3M | 1000h | 2 | 60.2 | 27.6 | 27.3 | 27.1 |
| CBN | LinLin | 3M | 1000h | 1 | 60.9 | 27.6 | 27.7 | 27.1 |
| CBN | LinLin | 3M | 1000h | 2 | 60.8 | 27.0 | 26.8 | 26.7 |

on 1000 hours of training data in a 3-day time, when considering 12 iterations ≈6 hours each, as shown in Tab. I. This is now possible with GPGPU training. Also, we have shown that replacing Sigmoidal bottleneck with Linear one leads to WER reduction, however such ANN is more difficult to train and initial learning rate has to be tuned. Finally, the idea of Convolutional Bottleneck Network was presented and studied. The optimal CBN has two linear bottlenecks and is trained with the shared-part pre-training.

The relative WER improvement compared to the previously published Universal Context ANN is **5.5%**. If we compare to the 5-layer single-bottleneck network of the same number of parameters, the relative improvement is **17.7%**. Both improvements refer to the systems with plain BN-features trained by Maximum Likelihood.

In the future, we will focus on tuning of the GMM-HMM part of the Tandem. First, it is possible to use more advanced language model, for example 4-gram or Recurrent Neural Network LM [10]. Also, the GMM-HMM model can be trained discriminatively by fMPE and speaker-adapted by CMLLR. Furthermore, it is still possible to improve the ANN model. Since we are dealing with a 7-layer deep architecture, it might be beneficial to pre-train the ANN by Restricted Boltzmann Machines [11]. Another very promising results were recently reported on the Deep Sparse Rectifier Neural Networks [12].

## ACKNOWLEDGMENT

This work was partly supported by Technology Agency of the Czech Republic grant No. TA01011328, Czech Ministry of Education project No. MSM0021630528, Grant Agency of Czech Republic project No. 102/08/0707 and No. 102/09/P635, and by Czech Ministry of Trade and Commerce project No. FR-TI1/034.

## REFERENCES

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

[2] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. ICASSP'00*, vol. 3, 2000, pp. 1635–1638.

[3] F. Grézl, M. Karafiát, S. Kontár, and J. Černocký, "Probabilistic and bottle-neck features for lvcsr of meetings," in *Proc. ICASSP'07*, 2007, pp. 757–760.

[4] F. Grézl and M. Karafiát, "Hierarchical neural net architectures for feature extraction in asr," in *Proc. INTERSPEECH'10*, 2010, pp. 1201–1204.

[5] K. Veselý, L. Burget, and F. Grézl, "Parallel training of neural networks for speech recognition," in *Proc. INTERSPEECH'10*, 2010, pp. 2934–2937.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed., ser. Information Science and Statistics. Springer, 2007.

[7] P. Schwarz, P. Matějka, and J. Černocký, "Towards lower error rates in phoneme recognition," *Lecture Notes in Computer Science*, vol. 2004, no. 3206, pp. 465–472, 2004.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation 9(8):1735-1780*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] T. Hain, V. Wan, L. Burget, M. Karafiát, J. Dines, J. Vepa, G. Garau, and M. Lincoln, "The ami system for the transcription of speech in meetings," in *Proc. ICASSP'07*, 2007, pp. 357–360.

[10] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. INTERSPEECH'10*, vol. 2010, 2010, pp. 1045–1048.

[11] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," in *Neural Computation*, vol. 18, 2006.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. AISTATS'10*, ser. W&CP, vol. 15 (draft). JMLR, 2010.