Tuning of Acoustic Modeling and Adaptation Technique for a Real Speech Recognition Task

Jan Vaněk, Josef Michálek(✉), and Josef Psutka

University of West Bohemia, Univerzitní 8, 301 00 Pilsen, Czech Republic
{vanekyj,orcus,psutka}@kky.zcu.cz

Abstract. At the beginning, we had started to develop a Czech telephone acoustic model by evaluating various Kaldi recipes. We had a 500-h Czech telephone Switchboard-like corpus. We had selected the Time-Delay Neural Network (TDNN) model variant "d" with the i-vector adaptation as the best performing model on the held-out set from the corpus. The TDNN architecture with an asymmetric time-delay window also fulfilled our real-time application constrain. However, we were wondering why the model totally failed on a real call center task. The main problem was in the i-vector estimation procedure. The training data are split into short utterances. In the recipe, 2-utterance pseudospeakers are made and i-vectors are evaluated for them. However, the real call center utterances are much longer, in order of several minutes or even more. The TDNN model was trained from i-vectors that did not match the test ones. We propose two ways how to normalize statistics used for the i-vector estimation. The test data i-vectors with the normalization are better compatible with the training data i-vectors. In the paper, we also discuss various additional ways of improving the model accuracy on the out-of-domain real task including using LSTM based models.

Keywords: Neural networks · Acoustic model · Automatic speech recognition · Adaptation · I-vectors

1 Introduction

Deep neural networks (DNNs) have been successfully applied to acoustic modelling for automatic speech recognition (ASR). ASR systems are now capable of real-world applications, especially if we have plenty of data from the target domain. However, there can be a performance degradation due to the mismatch between training and testing conditions, such as speaker, recording channel, speaking style, and acoustic environment [3,12]. Many approaches have been proposed in recent years to achieve a robust ASR or to improve the DNN adaptability. Generally, there are two ways that can also be combined [1,15]. First, a boost of the training data variability [6,7]. Second, an adaptation of the acoustic model [5,11,13]. Variability of the training data can be improved by an addition

C. Martín-Vide et al. (Eds.): SLSP 2019, LNAI 11816, pp. 235–245, 2019.
https://doi.org/10.1007/978-3-030-31372-2_20

of more sources of real speech or by an artificial modification of the existing data itself – usually called data augmentation. In this paper, we have evaluated these approaches on a real call center speech recognition task. We have identified that a proper use of the i-vector adaptation is crucial. Especially an i-vector statistics normalization. Therefore, we focus on these techniques in detail below.

2 I-Vector Calculation

We only outline the main points of the i-vector calculation here. More detail can be found in [4,11]. The acoustic feature vectors $\mathbf{x}_t \in \mathcal{R}^D$ are seen as samples generated from a universal background model (UBM) represented as a GMM with K diagonal covariance Gaussians

$$\mathbf{x}_t \sim \sum_{k=1}^{K} c_k \mathcal{N}(\cdot; \mu_k(0), \mathbf{\Sigma}_k) \tag{1}$$

with mixture coefficients c_k, means $\mu_k(0)$ and diagonal covariance matrices $\mathbf{\Sigma}_k$. Moreover, data $\mathbf{x}_t(s)$ belonging to speaker s are drawn from the distribution

$$\mathbf{x}_t(s) \sim \sum_{k=1}^{K} c_k \mathcal{N}(\cdot; \mu_k(s), \mathbf{\Sigma}_k) \tag{2}$$

where $\mu_k(s)$ are the means of the GMM adapted to speaker s. The basis of the i-vector algorithm is to assume a linear dependence between the speaker-adapted means $\mu_k(s)$ and the speaker-independent means $\mu_k(0)$ of the form

$$\mu_k(s) = \mu_k(0) + \mathbf{T}_k \mathbf{w}(s), \qquad k = 1 \ldots K \tag{3}$$

$\mathbf{T}_k$, of size $D \times M$, is called the factor loading submatrix corresponding to component k and $\mathbf{w}(s)$ is the speaker identity vector (i-vector) corresponding to s. Each $\mathbf{T}_k$ contains M bases which span the subspace with important variability in the component mean vector space.

For the i-vector estimation, we assume a fixed soft alignment of frames to mixture components. We estimate the posterior distribution of $\mathbf{w}$ given speaker data as

$$p(\mathbf{w}|\mathbf{x}_t(s)) = \mathcal{N}(\mathbf{w}; \mathbf{L}^{-1}(s) \sum_{k=1}^{K} \mathbf{T}_k^T \mathbf{\Sigma}_k^{-1} \mathbf{\Theta}_k(s), \mathbf{L}^{-1}(s)) \tag{4}$$

with precision matrix $\mathbf{L}(s)$ of size $M \times M$ expressed as

$$\mathbf{L}(s) = \mathbf{I} + \sum_{k=1}^{K} \gamma_k(s) \mathbf{T}_k^T \mathbf{\Sigma}_k^{-1} \mathbf{T}_k \tag{5}$$

The quantities that appear in (4) and (5) are the zero-order and centered first-order statistics and are defined as

$$\gamma_k(s) = \sum_t \gamma_{kt}(s), \tag{6}$$

$$\Theta_k(s) = \sum_t \gamma_{kt}(s)(\mathbf{x}_t(s) - \mu_k(0)) \tag{7}$$

with $\gamma_{kt}(s)$ being the posterior probability of mixture component k given $\mathbf{x}_t(s)$. The i-vector that we are looking for is simply the MAP point-estimate of the variable $\mathbf{w}$ which is the mean of the posterior distribution from (4), i.e.

$$\mathbf{w}(s) = \mathbf{L}^{-1}(s) \sum_{k=1}^{K} \mathbf{T}_k^T \mathbf{\Sigma}_k^{-1} \mathbf{\Theta}_k(s). \tag{8}$$

The inversion of matrix $\mathbf{L}^{-1}(s)$ could be numerically unstable. More robust variant to i-vector estimation is based on a linear solver. E.g. Kaldi implementation uses a linear conjugate gradient solver.

Because of nature of the MAP point-estimate, low amount of accumulated data leads to an i-vector close to the central zero point. A DNN trained from short utterances works with i-vectors that are not far from the central zero point. Long utterances in the test set produce precise i-vectors far from the central zero point that the DNN never saw. In that case, the long test utterance i-vectors cause failure of the recognizer. Also, we have investigated the online scenario, where first several words of the utterance are recognized well, but when the utterance length starts to be significantly longer than the typical training length, recognition errors ramp up. The solution is an i-vector statistics normalization comparable with the training utterances length distribution.

3 I-Vector Statistics Normalization Methods

The most efficient method of i-vectors normalization is to scale statistics (Eqs. (5), (6), and (7)) before the i-vector calculation because of the additive nature of statistics and real meaning of them (amount of accumulated data in seconds).

3.1 Length Normalization

The simplest way is to scale down statistics to some predefined length of data in seconds. The proper length may be derived from training utterance (pseudospeaker) lengths. Disadvantage of this approach is a convergence to a constant i-vector for very long utterances. This i-vector calculated from long-term statistics is not precisely compatible with the short-term i-vectors in training data.

3.2 Exponential Forgetting

To focus more on the local short-term information, a local time-window may be used. Similar, but simple to implement, is the exponential forgetting of the statistics. Every time-step, the statistics accumulators are multiplied by a constant less than one. The constant value α_e is set compatible to a time-window length T_w, thus easy to understand

$$\alpha_e = 1 - \frac{1}{T_w}, \tag{9}$$

where T_w is the time-window length in time-step units, e.g. frames.

4 Experiments

4.1 Training Data

As a main source of our training data, we have a 500-h Czech telephone Switchboard-like corpus. It is called *Bezplatne Hovory (BH)* (eng. Toll-free Calls). The corpus consists of unrestricted spontaneous speech in variable conditions and speaking styles. The training data part was filtered a bit, we omitted very short utterances and utterances with majority of non-speech events. The total length of data for training was 406.6 h. The total number of calls was 5,535. It does not match perfectly with the number of speakers because some speakers may call more than once.

To add more variability and robustness into the training data, we added three additional corpora:

- *Siemens* – a read speech corpus recorded through a telephone channel. It is a small corpus of 10.7 h of speech in total, but with a large number of speakers: 1,121.
- Czech part of *SpeechDat-E (SD-E)* – a read speech telephone corpus [8]. The Czech part used for training consists of 739 speakers and 20.8 h of speech in total.
- *Telephone Quality Speech Corpus (TQSC)* – a read speech telephone corpus recorded at our department. Each of 1,929 speakers uttered 40 sentences. These sentences were uttered by native Czech male and female speakers, and they contain a large number of silent parts and low–level noises. The used training data has 26.2 h in total.

All added corpora are read speech ones, because we were interested in testing whether the addition of a read speech into the training data improves the spontaneous speech recognition results.

4.2 Test Data and Recognizer Setup

We have prepared two tests. First, an in-domain test called *BH Test*. The data for this test were taken from a held-out part of the main training corpus (BH). We have selected 221 utterances of 26 speakers. The test data has 17.3 min in total. The in-domain test showed how an acoustic model performed in the matching conditions.

Second, an out-of-domain test from a real call center task. We have split an Operator and Customer part of the test for the evaluation. Operators are skilled speakers with more formal speech in contrast with more spontaneous customers. We selected 20 operator and 20 customer calls/utterances. The calls were about several minutes long and both groups were balanced to have 1.2 h of speech in total.

We have used our proprietary recognizer optimized for online speech recognition [10]. We have used two trigram language models. A general telephone conversation model with 550k words for the *BH Test* and a task-specific 33k-words model for the Call center test. OOV words were added explicitly to the vocabulary to avoid disturbing the recognition results.

4.3 Acoustic Model Training

Two kinds of acoustic models were trained.

TDNN Models. were trained in Kaldi [9,14]. The procedure followed a Kaldi Switchboard example recipe. In our prior work, we tested various recipes and setups. Note that not every model architecture is usable for a real-time recognition. Therefore, we are restricted with absence of any offline technique and limited by the total latency. From models that are real-time recognition compatible, we have selected the "s5c" TDNN Switchboard recipe "tdnn_d".

The model is trained in three stages. First, a base GMM-HMM is trained. The triphone clustered states (senones) and alignments produced by the GMM-HMM were then used for further TDNN training. Low resolution MFCC features were used for the entire GMM-HMM training. The GMM-HMM was trained with LDA, MLLT, SAT techniques. Second and third stages used high resolution MFCC features.

The second stage was estimation of an i-vector model and extraction of i-vectors for the entire training set. The i-vector model was based on pseudospeakers that were made by setting two utterances from a speaker as a pseudospeaker data. It boosted the speaker variability and enriched the i-vectors space. For even more speaker-space variability, a speed perturbation with 0.9 and 1.1 speed ratios was used [6]. Thus, the total amount of training data was tripled to almost 1400 h.

The third stage was the TDNN training itself. The TDNN has 6 hidden layers with 1024 ReLU neurons. The time-delay coefficients were as follows: (−2, −1, 0, 1, 2), (−1, 2), (−3, 3), (−3, 3), and (−7, 2). The total delay of the network is 12 frames. The final softmax layer has 7,149 outputs – senones – triphone states. The net was trained by Kaldi GPU parallel implementation with momentum SGD.

LSTM Models. follow the first two stages of the TDNN training [2]. They share the same features, triphone states, alignments, and i-vectors. Only a NN architecture and training procedure were different. We used Chainer 5.0 as a NN training framework for all LSTM models.

During our preliminary tests, we had found out that a residual neural network (ResNet) with LSTM layers worked best for our data. The neural network architecture used in this paper was 6-layer LSTM network with 1024 units in each layer. The skip connections were between 4-th and the last (linear) layer (see Fig. 1). First, we trained each network using Adam optimizer and then we

performed 3 training stages with the momentum SGD with momentum equal to 0.9 and learning rate equal to $1e-3$, $1e-4$, and $1e-5$, respectively. In all training stages, we used batch size 128 and dropout $p = 0.2$. We used early stopping, where each training stage was stopped when a validation data criterion increased in comparison to the last epoch.

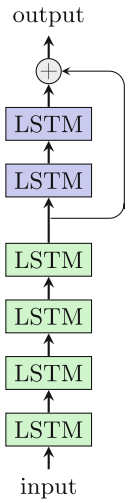

Fig. 1. Our LSTM ResNet architecture

4.4 Results

First, we evaluated the impact of i-vectors on the TDNN network performance. We trained several TDNN models. The models were trained on the BH only dataset and on a combined datasets BH, Siemens, SD-E, and TQSC (called "All" in the figures). We also trained each TDNN model with and without i-vectors. The models with i-vectors used exponential forgetting with α_e equivalent to 5s time window length. All TDNN models were trained on speed perturbed (SP) data with speed ratios 0.9 and 1.1.

From the Fig. 2 it can be seen, that models with i-vectors have worse WER than models without i-vectors on the in-domain test. However, on the out-of-domain tests, models with i-vectors perform better. The biggest impact of adding i-vectors on the in-domain test was for model trained on BH only, the resulting WER increased by 1.39%. The biggest improvement on out-of-domain tests was for model trained on All datasets, where call center customer WER improved by 3.37%. The use of All datasets compared to only BH had negligible impact on models without i-vectors, but on models with i-vectors it noticeably improved WER for all tests. The biggest improvement was 1.03% lower WER on call center customer test.

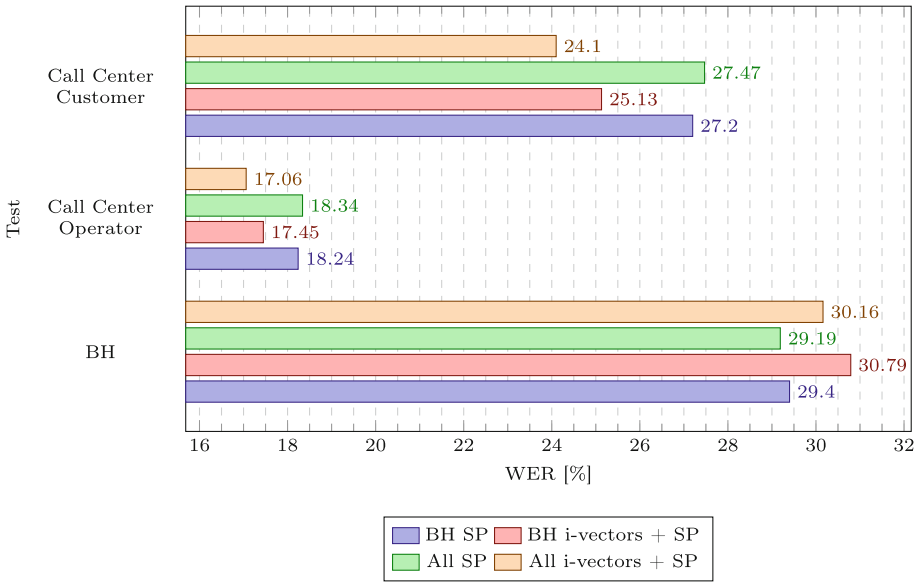

Fig. 2. Word Error Rate [%] of TDNN Models on In-domain and Out-of-domain Recognition Task

Then, we trained LSTM models as described in Sect. 4.3. As with TDNN, all LSTM models were trained on BH only dataset and also on combined datasets called "All". We also trained all models with and without i-vectors. And we trained all models with and without speed perturbation (SP).

The results for LSTM models are in Fig. 3. From the figure, it is obvious that for all tests, in-domain and out-of-domain, all models with i-vectors perform worse than models without i-vectors. The biggest difference in WER with i-vectors compared to without is for a model trained on All datasets without SP, where the call center customer test WER got worse by even 17.96%. Speed perturbation improved results for all test cases except for models trained on BH with i-vectors. In-domain test in this case has lower WER, but out-of-domain tests have higher WER, with call center customer WER increasing by 2.5%. The best performing model for out-of-domain call center operator test was a model trained on All datasets with SP without i-vectors, having 17.36% WER. The best models for other out-of-domain test (customer) and in-domain test were both trained on BH only with SP without i-vectors, but their WER were very close in comparison to the best model for call center operator test mentioned above.

Next, we evaluated the performance of two i-vector normalization techniques. When we started working with TDNN models with i-vectors, we found out that the models performed well on in-domain test, but failed on out-of-domain tests. We found out, that the reason was an utterance length. Our out-of-domain test (real call center data) contained utterances much longer than our training data and some form of i-vector normalization had to be employed.

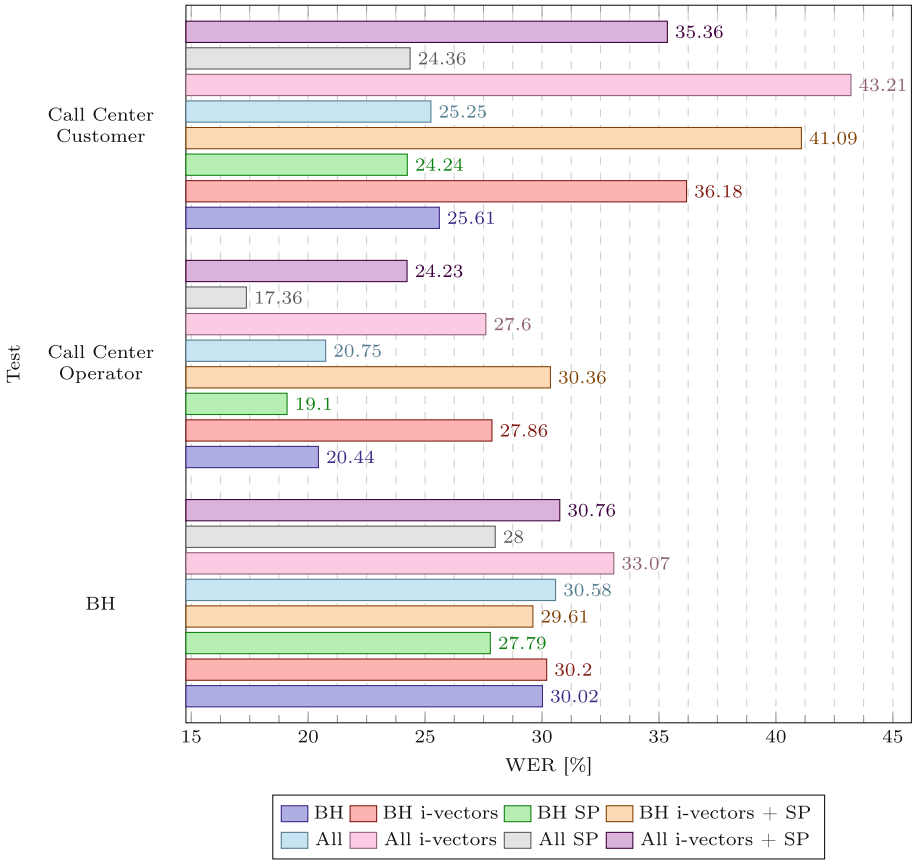

Fig. 3. Word error rate [%] of LSTM models on in-domain and out-of-domain recognition task

The image Fig. 4 shows a histogram of a pseudospeaker utterance length for our training and out-of-domain test data. It can be seen that training pseudospeaker data are generally up to 10 s (typically 3–5 s) long while test utterances are mostly longer than one minute.

We have evaluated two i-vector normalization techniques, the exponential forgetting and the length normalization as described in Sect. 3. We have trained a model on various normalized data lengths and evaluated WER of each normalization technique. Used model was TDNN trained on all datasets with SP. The results can be seen in Fig. 5. From the figure it can be said that the best results for all the normalization techniques are generally obtained using data length from 3 to 6 s. Length normalization gives best WER on both tests for data length of 3 s and exponential forgetting gives best WER on operator test for data length of 4 s and on customer test for data length of 6 s, although WER for all tests in this range are very similar. Exponential forgetting gives better

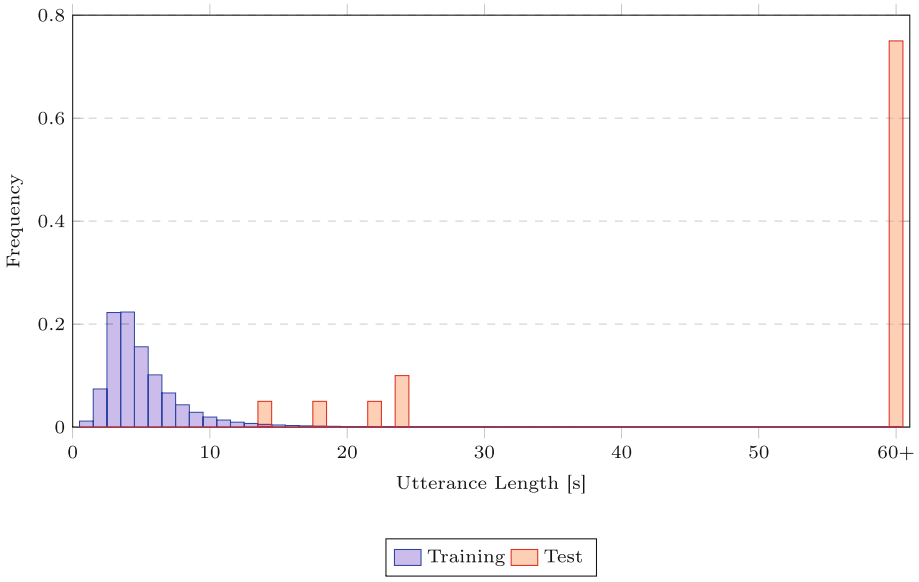

Fig. 4. Histogram of pseudospeaker utterance length in training and test data

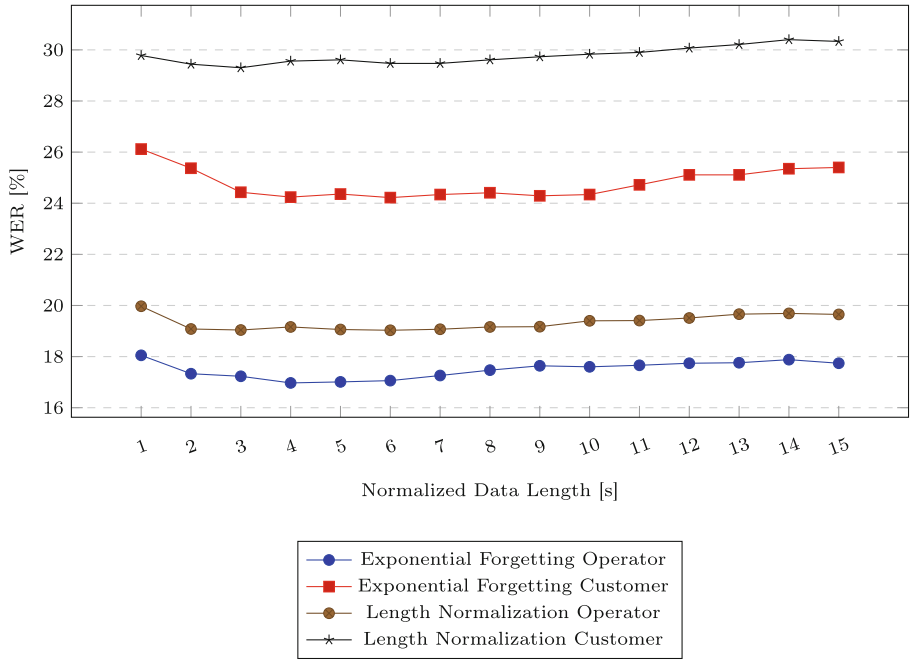

Fig. 5. Impact of data length on the word error rate [%] of TDNN models with different i-vector normalization techniques

WER than length normalization on all our tests for all normalized data lengths. The WER obtained using exponential forgetting compared to the length normalization is improved in average by 7.3% for operator test and by 10.4% for customer test.

5 Conclusion

In this paper, we described our experiences with the development of the Czech telephone acoustic model. We have tested models based on two architectures, TDNN and LSTM, suited for the real-time recognition task. At first, the models totally failed on a real call center task. We have identified the main problem in the i-vector estimation procedure and propose and evaluate two i-vector statistics normalization methods. The use of the exponential forgetting compared to the length normalization was far better. The forgetting constant α_e was robust to set and a value matching a typical training pseudospeaker data length worked well.

We also tested various additional techniques: data augmentation, addition of real data, and i-vector adaptation. Generally, we may recommend using the speed perturbed data augmentation. Other techniques behaviour was model architecture dependent. TDNN models worked well with i-vectors on the call center test and addition of the out-of-domain real data did not help. In contrast, with LSTM models, the i-vector adaptation failed, speed perturbation helped in all cases, and adding the read speech data helped only on the call center operator test.

Summarized, LSTM based model worked better by more than 1% absolutely than TDNN on the in-domain test. In contrast, the TDNN model with the proper i-vector normalization was more robust and worked slightly better on the out-of-domain test. It seems, that the LSTM model has an ability to outperform the TDNN one. However, some other adaptation technique needs to be developed to improve its robustness on the out-of-domain data.

Acknowledgement. This research was supported by the LINDAT/CLARIN, project of the Ministry of Education of the Czech Republic No. CZ.02.1.01/0.0/0.0/16_013/0001781.

References

1. Chen, X., et al.: Recurrent neural network language model adaptation for multi-genre broadcast speech recognition. In: Sixteenth Annual Conference of the International Speech Communication Association (2015)
2. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
3. Hsu, W.N., Zhang, Y., Glass, J.: Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In: 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 16–23. IEEE (2017)
4. Karafiát, M., Burget, L., Matějka, P., Glembek, O., Černockỳ, J.: iVector-based discriminative adaptation for automatic speech recognition. In: 2011 IEEE Workshop on Automatic Speech Recognition & Understanding, pp. 152–157. IEEE (2011)

5. Karanasou, P., Wang, Y., Gales, M.J., Woodland, P.C.: Adaptation of deep neural network acoustic models using factorised I-Vectors. In: Fifteenth Annual Conference of the International Speech Communication Association (2014)
6. Ko, T., Peddinti, V., Povey, D., Khudanpur, S.: Audio augmentation for speech recognition. In: Sixteenth Annual Conference of the International Speech Communication Association (2015)
7. Park, D.S., et al.: Specaugment: a simple data augmentation method for automatic speech recognition. arXiv preprint arXiv:1904.08779 (2019)
8. Pollak, P., et al.: SpeechDat(E) - Eastern European telephone speech databases. In: Proceedings LREC 2000 Satelite workshop XLDB - Very Large Telephone Speech Databases, pp. 20–25. European Language Resources Association, Athens (2000)
9. Povey, D., et al.: The kaldi speech recognition toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society (Dec 2011), iEEE Catalog No.: CFP11SRW-USB
10. Pražák, A., Müller, L., Šmídl, L.: Real-time decoder for LVCSR system. In: The 8th World Multi-Conference on Systemics, Cybernetics and Informatics: vol. VI : Image, Acoustic, Signal Processing and Optical Systems, technologies and applications, pp. 450–454. International Institute of Informatics and Systemics, Orlando, Florida (2004). http://www.kky.zcu.cz/en/publications/PrazakA_2004_Real-timedecoderfor
11. Saon, G., Soltau, H.: Speaker adaptation of neural network acoustic models using I-Vectors. In: IEEE Workshop on Automatic Speech Recognition and Understanding pp. 55–59 (2013). https://doi.org/10.1109/ASRU.2013.6707705,http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6707705
12. Sim, K.C., et al.: Domain adaptation using factorized hidden layer for robust automatic speech recognition. In: Proceedings of the INTERSPEECH (2018)
13. Sim, K.C., Qian, Y., Mantena, G., Samarakoon, L., Kundu, S., Tan, T.: Adaptation of deep neural network acoustic models for robust automatic speech recognition. In: Watanabe, S., Delcroix, M., Metze, F., Hershey, J.R. (eds.) New Era for Robust Speech Recognition, pp. 219–243. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64680-0_9
14. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.J.: Phoneme recognition using time-delay neural networks. Backpropagation: Theory, Architectures and Applications, pp. 35–61 (1995)
15. Yu, D., Li, J.: Recent progresses in deep learning based acoustic models. IEEE/CAA J. Automatica Sin. **4**(3), 396–409 (2017)