Deep Neural Networks in Russian Speech Recognition

Nikita Markovnikov[1,2(✉)], Irina Kipyatkova[2], Alexey Karpov[2], and Andrey Filchenkov[1]

[1] ITMO University, Saint-Petersburg, Russia
niklemark@gmail.com
[2] SPIIRAS Institute, Saint-Petersburg, Russia

Abstract. Hybrid speech recognition systems incorporating deep neural networks (DNNs) with Hidden Markov Models/Gaussian Mixture Models have achieved good results. We propose applying various DNNs in automatic recognition of Russian continuous speech. We used different neural network models such as Convolutional Neural Networks (CNNs), modifications of Long short-term memory (LSTM), Residual Networks and Recurrent Convolutional Networks (RCNNs). The presented model achieved 7.5% reducing of word error rate (WER) compared with Kaldi baseline. Experiments are performed with extra-large vocabulary (more than 30 h) of Russian speech.

Keywords: Deep learning · Russian speech · Speech recognition
Acoustic models

1 Introduction

Automatic speech recognition (ASR) is a process of converting speech to text. It can be performed using both acoustic model (AM) and language model (LM) as shown in [1]. In this paper, we consider building and learning of acoustic models only.

Acoustic models are traditionally built using hidden Markov models (HMM) with the Gaussian mixture model (GMM). However, hybrid deep neural networks with Hidden Markov Models (DNN-HMM) models recently showed better results and reduced error of speech recognition [2].

DNN models for languages with strict word order (e.g. English) suit well, but as for the Russian language, these models are not such efficient. Our motivation is to find neural network architecture that would accomplish an improvement of our Kaldi baseline.

Recently, there some promising models were proposed. For instance, recurrent neural networks such as the long short-term memory have achieved significant results in speech recognition. However, LSTMs are easy to overfit. Convolutional neural networks (CNNs) is a popular class of deep neural networks, but it has

A. Filchenkov et al. (Eds.): AINL 2017, CCIS 789, pp. 54–67, 2018.
https://doi.org/10.1007/978-3-319-71746-3_5

not achieved large reduction of recognition error. Our goal is to construct and apply various deep neural networks to the problem of automatic recognition of continuous Russian speech.

To study an acoustic model, we need a large corpus of the Russian speech. In this work, neural networks were constructed using extra-large vocabulary with more than 25 h of the Russian continuous speech that will be described below.

The performance of our ASR systems was evaluated in term of word error rate (WER). This metric is computed using the Levenstein distance between the recognized sequence and the truth sequence and it is expressed in percentage as follows:

$$WER = \frac{D + S + I}{N} \cdot 100\%$$

where N denotes the total number of words in the truth sequence, D is the number of deletions, S is the number of substitutions and I denotes the number of insertions.

The rest of the paper is organized as follows. In Sect. 2, we survey related works. In Sect. 3, we describe architectures of DNNs that we used for the constructing AMs. In Sect. 4, we discuss datasets for a training and testing AMs and our language model. In Sect. 6, we describe our experimental setup and present configurations of neural networks and the results. Finally, we conclude in Sect. 7.

2 Related Work

We give a brief overview of Kaldi [3]. Kaldi is a toolkit written in C++, integrated with OpenFST toolkit for a support of finite state transducers. Also, it uses BLAS and LAPACK libraries for a support of linear algebra operations. Kaldi purpose is to have a modern and flexible code, since it is easy to be extended and modify. Kaldi is an open-source toolkit and it is available for modifications. Kaldi provides two realizations of neural network training. The first one is Kerel's implementation [4] that supports pretraining using deep belief networks and training using GPU. The second realization is Dan's implementation [5] that does not use pretraining, but provides parallel training using several CPUs.

A speech recognition system for the Italian speech on CHILDIT corpus was suggested in [6] using Kaldi toolkit. The best result was shown by a hybrid system with deep neural networks. Kaldi demonstrated the effectiveness of easily usage of DNN in order to reduce recognition error comparing with other toolkits for automatic speech recognition. Kaldi provides a baseline for speech recognition.

A system for recognition Serbian speech was described in [7]. Serbian is in the same language group as Russian, thus it is interesting for us. The system was written using CUDA. System performance was examined using Kaldi. WER for HMM/GMM was 63.39%, while for a hybrid system with deep neural network it was 48.5% resulting into improvement by 15–22% in dependence on testing data.

Also, system for Russian speech recognition was described in [8]. Modeling was performed using deep neural network and studying was provided with GPU.

There were described two types of recognition. The first model used features that were got in the bottleneck and the second model used a hybrid approach with neural network. Baseline was 31.5% and the best result was 25.1%.

A research on hybrid models for Russian speech recognition was presented in [9]. Various configurations of neural networks were learned with various numbers of layers, their dimensions as well as with various activation functions including hyperbolic tangent and p-norm. For a constructing of acoustic models and testing Kaldi toolkit was used. Experiments were performed using acoustic models built with GMM and hybrid models with DNNs. Baseline was 25.32% and the best result was 20.3%, so a reduction of an error was approximately 20%.

3 Architectures of Neural Networks for Acoustic Modeling

In this section we will shortly describe architectures of neural networks that we used for the experiments.

3.1 LSTM

Standard LSTM. LSTM network [10] consists of several LSTM-units which are chained consequentially. LSTM-units have several gates that control data flow for saving and removing from the unit. One of such gates, forget gate layer is

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f),$$

where h_{t-1}, x_t are inputs, σ is the logistic activation function and f_t is an output value between 0 and 1.

Then, LSTM-unit uses layer that filters data for saving. It consists of two parts. The first one is an input gate layer:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i).$$

And the second one is

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C).$$

Thus, a value of the new state is computed in the following way:

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t.$$

Peephole LSTM. Peephole LSTM is a modification of the standard LSTM. Its gates are allowed to look at a state of a LSTM-unit. So, gates are:

$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f);$$

$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i);$$

$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o).$$

BLSTM. One disadvantage of the standard LSTMs is an opportunity to use only previous context. Looking at the future context may be useful in speech recognition of a language with complex grammar as Russian. This aspect is included in bidirectional recurrent neural network (BRNN). They can be described as follows:

$$\overrightarrow{h}_t = \sigma(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}});$$

$$\overleftarrow{h}_t = \sigma(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}});$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y.$$

A combination of BRNN and LSTM gives bidirectional LSTM (BLSTM).

3.2 CNN

The next wide class of neural networks is convolutional neural networks (CNN) [11]. These models consist of layers of three types: convolutional layers, subsampling layers and fully-connected layers. The main idea of CNNs is increasing the density of uncorrelated sections of the features.

Discrete convolution operation is used for building convolutional layers and written as

$$(f * g)[n] = \sum_m f[m]g[n-m],$$

where f is a feature matrix and g is a convolution kernel.

The output of neurons can be represented as

$$h^l = f(h^{l-1} * k^l + b^l),$$

where h^l is an output vector of lth layer, f is an activation function, b is a bias and k is convolution kernel. The result is called feature map.

Subsampling layers reduce dimension of input feature maps. They are divided into several types such as max-pooling, average-pooling, etc.

3.3 ResNet

A deep convolutional residual network was presented for image recognition in [12]. Its main component is a residual unit:

$$y_l = h(x_l) + F(x_l, W_l), x_{l+1} = f(y_l),$$

where x_l and x_{l+1} are an input and an output of the lth unit, F is a residual function.

In that paper, $h(x_l) = x_l$ and f was ReLU. The main idea is to learn the residual function F. F can use some activation function, convolutional layers, etc. Residual unit is presented in Fig. 1.

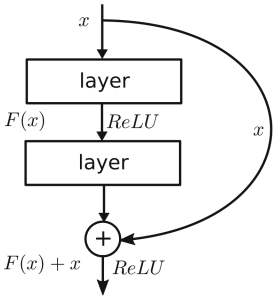

Fig. 1. Residual unit

3.4 RCNN

In papers [13,14], a combination of RNN and CNN was proposed. It was called recurrent convolutional neural network (RCNN). That model was used for object recognition and scene labeling. The main unit is a recurrent convolutional layer (RCL):

$$h_t(i,j) = \sigma\left(\sum_{i'=-s}^{s}\sum_{j'=-s}^{s} w_k^f\,(i',j')\,x_t\,(i-i',j-j') + \sum_{i'=-s}^{s}\sum_{j'=-s}^{s} w_k^r\,(i',j')\,h_{t-1}\,(i-i',j-j') + b\right),$$

where w_k^f and w_k^r are kernels.

$\sigma(x) = f(g(x))$ is a superposition of two functions. $g(x)$ can be the sigmoid function or ReLU. $f(\cdot)$ is a normalization function. Batch-normalization [15] can be used as a normalization function to speed up the learning process. There are T time steps. Network depth grows up with growth of T. Also, it can be extended with max-pooling and other layers. Schema of RCNN is presented in Fig. 2.

4 Datasets

4.1 Dataset for the Acoustic Models

In this work, we use the training speech corpus collected at SPIIRAS as in [9] and combined using three databases:

- recordings of 50 native Russian speakers, 16,350 utterances. Each speaker pronounced a set of 327 phrases;
- recordings of 55 native Russian speakers where each speaker pronounced 105 phrases;

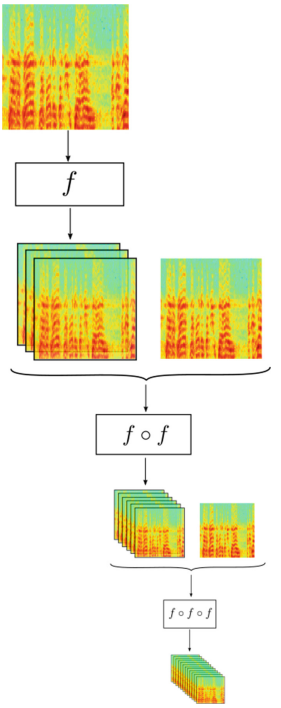

Fig. 2. Recurrent convolutional network ($T = 3$), where f is a convolutional layer

– the third part is an audio part of the audio-visual speech corpus HAVRUS [16]. 20 native Russian speakers (10 male and 10 female speakers) with no language or hearing problems participated in the recordings. Each of them pronounced 200 Russian phrases.

The total duration of the entire speech corpus is more than 30 h.

To test the system, we use a speech database of 500 phrases pronounced by 5 speakers. The phrases were taken from the materials of Russian online newspaper "Fontanka.ru"[1] that was not used in the training data.

4.2 Dataset for the Language Model

Language model is an important part of the recognition system. Our language model is learned using data from a Russian news sites [17]. Dataset for the training of language model contains approximately 300 millions of collocations. As a language model n-gram ($n = 2$) model with KneserNey smoothing [18] is used.

[1] http://www.fontanka.ru/.

5 Speech Recognition System Implementation

For building and testing acoustic models, Kaldi toolkit [3] was used.

We need to choose a toolkit for a studying and configuring neural networks. Popular toolkits are Theano, Caffe, Torch, TensorFlow, CNTK, MXNET, Deeplearing4j, etc. There are several papers comparing these toolkits: [19–22]. As a result, CNTK was chosen because it has several advantages in comparison with TensorFlow:

- clear and simple network description using BrainScript or NDL,
- simple realization of combining with Kaldi (lesser number of code lines),
- short description of LSTMs and CNNs,
- a lot of examples,
- wide support of using GPUs.

We used BrainScript for configuring neural networks. Kaldi's features were read with CNTK's Kaldi2Reader module. SGD with learning rate per minibatch 0.1 was used with size of minibatch 256. All experiments were provided using NVIDIA GeForce GT 730M.

6 Experiments and Results

In this section, we describe neural networks configurations and experiments on using them for continuous Russian speech recognition.

6.1 Baseline

Baseline is implemented using standard Kaldi steps as in [9]. Firstly, we extract features (13 mel-frequently cepstral coefficients [23]) from the training and the testing speech datasets. Then, we learn and tune monophone acoustic models. After that, we learn a triphone model using previous models. Finally, LDA, MLLT [24], SAT [25] and fMLLR [26] is applied.

The final step was studying hybrid DNN-HMM model. It takes 440 input features after LDA application. Neural network is a multi-layer perceptron (MLP) that consists of four hidden layers with tanh activation function ending with soft-max layer. Also, weight matrix initialization using DBN [27] is applied.

The baseline achieves 23.96% of WER.

Also, we compared our models' results with model from [8]. It used *nnet3* Kaldi's configuration that applied BLSTMs for speech recognition. The following configuration of the network was applied: three forward and three backward layers, 1024 cell and hidden dimensions, 128 recurrent and nonrecurrent projection dimensions. An initial learning rate was 0.0003 and final learning rate was 0.00003. This model achieves 22.8% of WER.

Table 1. Results for MLPs

	1 model	2 model	3 model	4 model
Layers	3	6	6	6
Dimensions	450×3	2048×6	512×6	512×6
Activation function	sigmoid	sigmoid	tanh	p-norm ($p = 2$)
Iterations	20	20	18	18
WER	25.54%	25.32%	24.96%	24.26%

6.2 MLP

Firstly, experiments on MPLs with various activation functions were provided using CNTK. We test four configurations presented in Table 1 together with their results. The fourth model showed the best result and it used p-norm activation function as in [9]:

$$y = \|x\|_p = \left(\sum_i |x_i|^p \right)^{1/p}.$$

We test models with $p = 1$ and $p = 3$, but the best result was made with $p = 2$.

6.3 LSTM

Configurations and results of applying LSTMs are presented in Table 2. The best result is shown by BLSTM. This model surpassed result of the baseline. However, a disadvantage of LSTM is that it can easily get overfitted and a lot of computational time is required to tune proper parameters.

Table 2. Results for LSTMs

	LSTM	PLSTM	BLSTM
Layers	6	3	3
Dimensions	512×6	512×3	512×3
Iterations	16	14	10
WER	23.32%	24.12%	23.08%

6.4 CNN

To learn CNN, we transformed features into tensors of dimension 40×11, where the first dimension is the time and the second one is the frequency.

The first model is a standard CNN. In the beginning, it has a convolutional layer with 64 output channels, 3×3 kernel, no padding and ReLU activation

function. Then, it has a max-pooling layer with 2×2 kernel, with padding and $2 : 2$ stride. Then, the same convolutional layer as the first one is applied, but with 128 output channels. After that, a max-pooling layer is used. Finally, two MLPs (dimension is 4096) with ReLU activation function are used.

WER achieved by CNN is 24.96%.

Standard CNN does not show a good result because of a degradation of the network. So, in [12] this problem was solved using residual units.

6.5 ResNet

We use ResNet architecture presented in Fig. 3 there is an architecture of ResNet that was used. After four iterations it receives WER = 22.17%. This result improved the baseline by 7.5%.

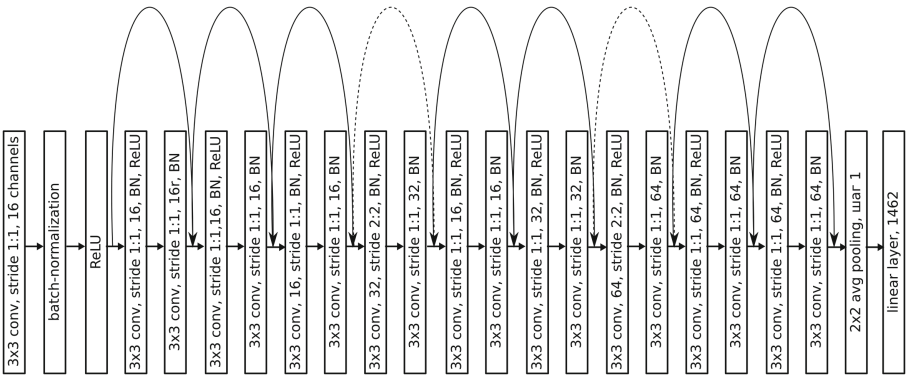

Fig. 3. ResNet

6.6 RCNN

Firstly, features are transformed into tensors of a dimension 40×11 and are sent to RCL stack input. RCL stack has the depth $T = 3$. The first convolution transforms data into 64 channels with padding, $2 : 1$ stride and 10×2 kernel. Then, the batch-normalization and ReLU are applied. The next convolution transforms the result of ReLU application into 4096 channels with padding and 3×1 kernel. After that, the batch-normalization and ReLU are applied to the sum of the previous and the current convolutions. Then, the same RCL-unit is applied. Convolutional layer with ReLU, 16×2 kernel and $1 : 1$ stride is applied to RCL stack output. Finally, three hidden sigmoid layers of 128 dimensions are applied with the batch-normalization.

After 12 iterations, we achieve WER = 22.56%. This result improves the baseline by 5.8%.

6.7 Comparing of Models

The best results shown by BLSTM, ResNet and RCNN are presented in Table 3. The speed of a decoding and a training is presented in Table 4. ResNet has shown the best result, but it was the slowest model. RCNN was faster, but it had a higher error of recognition.

Table 3. The best results of all models

	BLSTM	ResNet	RCNN
WER	23.08%	22.17%	22.56%

Table 4. Average speed of a training (features per second) and a decoding (utterances per second)

Model	Train	Decode
BLSTM	450.7	0.211
ResNet	121.4	0.105
RCNN	325.1	0.162

6.8 New Model

ResNet has shown the best result, but it was the slowest model. RCNN was faster, but it had a higher error of a recognition. LSTMs are difficult to be learned. Since, we can increase the density of uncorrelated sections of the features, simplify input features for the next studying using LSTMs. But CNNs show degradation, so we can use RCNNs and residual units.

Features were transformed into tensors of a dimension 40×11 and were sent to RCNN with $T = 3$. Then, there was a unit that was consist of two convolutional layers with a batch-normalization and ReLU, 3×3 kernel with padding and $1 : 1$ stride. Then, convolutional layer with 2×2 and $1 : 1$ stride. Finally, BLSTM's stack (three layers with 512 units in each layer) was applied.

That model gave WER = 22.34%. Also, other variations were applied. So, with $T > 3$ recognition error was growing up. A batch-normalization increased an error slightly, but it improved the training speed. So, the decoding speed was 0.134 utterances per second and the training speed 227.6 features per second. Also, a replacing of the last convolutional layer by max-pooling decreased error and it became 22.28%. But after an adding a residual unit we got a result WER = 22.07%, this result improved the baseline by 7.8%. The model is shown in Fig. 4.

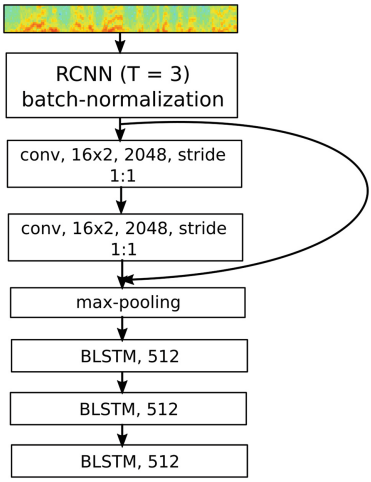

Fig. 4. RCNN and BLSTM union

Table 5. Results

Model	WER
Kaldi baseline	26.62%
Kaldi + DBN baseline	23.96%
Kaldi nnet3	22.80%
MLP-3-sigmoid	25.54%
MLP-6-sigmoid	25.32%
MLP-6-tanh	24.96%
MLP-6-p-norm	24.26%
LSTM	23.32%
PLSTM	24.12%
BLSTM	23.08%
CNN	24.92%
RCNN	22.56%
ResNet	**22.17%**
RCNN + CL + BLSTM	22.34%
RCNN + max-pooling + BLSTM	22.28%
RCNN + residual unit + max-pooling + BLSTM	**22.07%**

6.9 Summarization

The results of all discussed models that we used are shown in Table 5. So, ResNet and RCNN showed good results. A reduction of a recognition error was approximately 7.5%.

7 Conclusion

In this work, we consider the task of Russian speech recognition using hybrid DNN-HMM acoustic models. We used Kaldi and CNTK toolkits.

We used various neural network architectures: multilayer perceptron, LSTMs and theirs modifications, convolutional networks, residual convolutional networks and recurrent convolutional networks. The best result was shown by residual convolutional networks. After four iterations WER was 22.17%.

In the future we will provide experiments on using residual units, union with other models like BLSTMs using score fusion, applying an augmentation of the data. Also, we can use models that we've got for other languages (e.g. English). Moreover, we are interested in applying end-to-end systems for Russian speech.

Acknowledgments. This research is partially supported by the Council for Grants of the President of the Russian Federation (project No. MK-1000.2017.8) and by the Russian Foundation for Basic Research (project No. 15-07-04322).

References

1. Benesty, J., Sondhi, M.M., Huang, Y.: Introduction to speech processing. In: Benesty, J., Sondhi, M.M., Huang, Y.A. (eds.) Springer Handbook of Speech Processing. Springer Handbooks, pp. 1–4. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-49127-9_1
2. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.R., Jaitly, N., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Sig. Process. Mag. **29**(6), 82–97 (2012)
3. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Silovsky, J.: The Kaldi speech recognition toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (No. EPFŁ-CONF-192584). IEEE Signal Processing Society (2011)
4. Vesel K., Ghoshal, A., Burget, L., Povey, D.: Sequence-discriminative training of deep neural networks. In: Interspeech, pp. 2345–2349 (2013)
5. Povey, D., Zhang, X., Khudanpur, S.: Parallel training of DNNs with natural gradient and parameter averaging (2014). arXiv preprint arXiv:1410.7455
6. Cosi, P.: A KALDI-DNN-based ASR system for Italian. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–5. IEEE (2015)
7. Popović, B., Ostrogonac, S., Pakoci, E., Jakovljević, N., Delić, V.: Deep neural network based continuous speech recognition for Serbian using the Kaldi toolkit. In: Ronzhin, A., Potapova, R., Fakotakis, N. (eds.) SPECOM 2015. LNCS, vol. 9319, pp. 186–192. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23132-7_23
8. Prudnikov, A., Medennikov, I., Mendelev, V., Korenevsky, M., Khokhlov, Y.: Improving acoustic models for Russian spontaneous speech recognition. In: Ronzhin, A., Potapova, R., Fakotakis, N. (eds.) SPECOM 2015. LNCS, vol. 9319, pp. 234–242. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23132-7_29

9. Kipyatkova, I., Karpov, A.: DNN-based acoustic modeling for Russian speech recognition using Kaldi. In: Ronzhin, A., Potapova, R., Németh, G. (eds.) SPECOM 2016. LNCS, vol. 9811, pp. 246–253. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43958-7_29
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
11. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time series. In: The Handbook of Brain Theory and Neural Networks, vol. 3361, no. 10. The MIT Press, Cambridge (1995)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
13. Liang, M., Hu, X.: Recurrent convolutional neural network for object recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3367–3375 (2015)
14. Liang, M., Hu, X., Zhang, B.: Convolutional neural networks with intra-layer recurrent connections for scene labeling. In: Advances in Neural Information Processing Systems, pp. 937–945. Morgan Kaufmann Publishers Inc., San Francisco (2015)
15. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift (2015). arXiv preprint arXiv:1502.03167
16. Verkhodanova, V., Ronzhin, A., Kipyatkova, I., Ivanko, D., Karpov, A., Železný, M.: HAVRUS corpus: high-speed recordings of audio-visual Russian speech. In: Ronzhin, A., Potapova, R., Németh, G. (eds.) SPECOM 2016. LNCS, vol. 9811, pp. 338–345. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43958-7_40
17. Kipyatkova, I.S., Karpov, A.A.: Automatic processing and statistic analysis of a news text corpus for a language model of a Russian language speech recognition system. Inf. Upravl. Sist. **4**(47), 28 (2010)
18. Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. In: Proceedings of the 34th Annual Meeting on Association for Computational Linguistics, pp. 310–318. Association for Computational Linguistics (1996)
19. Fox, J., Zou, Y., Qiu, J.: Software frameworks for deep learning at scale. Internal Indiana University Technical report (2016)
20. Kovalev, V., Kalinovsky, A., Kovalev, S.: Deep Learning with Theano, Torch, Caffe, Tensorflow, and Deeplearning4J: Which One is the Best in Speed and Accuracy? (2016)
21. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Zhang, Z.: Mxnet: a flexible and efficient machine learning library for heterogeneous distributed systems (2015). arXiv preprint arXiv:1512.01274
22. Bahrampour, S., Ramakrishnan, N., Schott, L., Shah, M.: Comparative study of deep learning software frameworks (2015). arXiv preprint arXiv:1511.06435
23. Ganchev, T., Fakotakis, N., Kokkinakis, G.: Comparative evaluation of various MFCC implementations on the speaker verification task. In: Proceedings of the SPECOM, vol. 1, pp. 191–194 (2005)
24. Gopinath, R.A.: Maximum likelihood modeling with Gaussian distributions for classification. In: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, pp. 661–664. IEEE (1998)

25. Anastasakos, T., McDonough, J., Makhoul, J.: Speaker adaptive training: a maximum likelihood approach to speaker normalization. In: 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1997, vol. 2, pp. 1043–1046. IEEE (1997)
26. Gales, M.J.: Maximum likelihood linear transformations for HMM-based speech recognition. Comput. Speech Lang. **12**(2), 75–98 (1998)
27. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cogn. Sci. **9**(1), 147–169 (1985)