NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS

as a manuscript

Grachev Artem

Methods of Recurrent Neural Network Compression for Natural Language Processing

Ph.D. Dissertation Summary

for the purpose of obtaining academic degree
Doctor of Philosophy in Computer Science HSE

Moscow — 2019

The PhD Dissertation was prepared at National Research University Higher School of Economics.

Academic Supervisor: **Ignatov Dmitry**, Candidate of Sciences, Associate Professor, National Research University Higher School of Economics.

Work characteristic

Actuality of the work.

During the last decade the sub-field of machine learning called deep learning [1; 2] has been rapidly developing. The main feature of this sub-field is a successful training of neural networks with complex and **deep** architectures. Computational approaches associated with deep learning brought several areas of computer science to a new level. The related research directions include complex problems such as image processing, natural language generation and understanding, and speech recognition. In many areas and specific tasks, approaches based on deep learning methods have become recognized as industrial solutions [3]. At the same time, new problems and tasks emerge [2;4], while their research prospects are not yet exhausted.

The success of neural networks is explained by several factors. The first one lies in theoretical possibility to approximate arbitrary functions using neural networks [5; 6]. The second factor is the rescaling simplicity of the whole learning process. The network learning technique called back propagation makes it easy to compute the gradient of the loss function by the model parameters and do it in parallel. Definitely, the third component is the growth of computing power. The scalability of learning processes would worth nothing if it were not for the ability to scale all the involved computations. The emergence of high-power graphical cards (also known as Graphical Processing Units or GPU) allows training of neural networks cheaply and quickly.

At the same time, neural networks have their shortcomings and peculiarities. For example, despite the fact that from a theoretical point of view neural networks are considered as a universal approximator, currently there is no constructive way to build neural networks for any specific tasks. There are no methods that allow just choosing topology (e.g., specifying the number layers and their size) of the neural network for a particular task. In practice, specialists have to select hyperparameters responsible for the network topology during the learning process.

Another important drawback becomes obvious when there is a need to use neural networks on any devices: for example, mobile phones, TVs, vacuum cleaners or even refrigerators. Here, among the target problems audio signal processing, object recognition, keyboard suggestions or e-mail prompts can be mentioned. Typically, such devices do not have powerful processors and large amount of memory. At the same time, neural networks are demanding for both.

This drawback makes the problem of compression of neural networks relevant for their placement on various devices[1][2][3].

In this dissertation study, we consider the problem of compression of neural networks in the context of natural language modeling; here, we work mainly with recurrent neural networks. This allows us to take into account several important characteristics of this class of problems and adapt our methods for their solution. At the same time, most of the described methods are applicable in a wider context, i.e. for arbitrary neural networks.

The basic formulation of the language modeling problem assumes prediction of the next word from a previous sequence of words or, in other words, from the left context. The very first methods for solving this problem were based on the direct storage of all possible options. At the moment of prediction, the most probable option was chosen or sampling was made from the saved distribution. This approach has problems associated with taking into account long-distance dependencies and the need to store all the options. For example, all possible sequences of length four for real-world dictionaries can not longer be stored in the memory of computers, while recurrent neural networks are partially capable of simulating long-distance dependencies and do not require direct memorization of all variants.

However, they are still too large for their widespread use on the aforementioned devices that are usually very demanding of memory and speed. A natural requirement for building language models is the need to keep a dictionary of several thousand words (for example, about 20 thousand words cover only 80% of the Russian language). In addition, it is often necessary to solve the problem of predicting the next word or, to put it in terms of machine learning, to solve the problem of classification into several thousand (or even tens of thousands) classes. Therefore, despite the fact that neural networks occupy much less space than simple statistical models, the problem of neural networks compression is relevant for their widespread use.

Modern compression methods include both simple methods such as pruning of matrices or quantization of the representation of numbers and more complex ones based on decompositions of matrices or Bayesian techniques. Among the works based on pruning of matrices and quantization papers [7; 8] can be noted. Methods based on matrix decompositions can be classified into several categories. First of all,

[1]https://os.mbed.com/blog/entry/uTensor-and-Tensor-Flow-Announcement/

[2]https://towardsdatascience.com/why-machine-learning-on-the-edge-92fac32105e6

[3]https://petewarden.com/2017/05/08/running-tensorflow-graphs-on-microcontrollers/

it is possible to decompose matrices into low-rank factors [9], thereby achieving a certain level of compression. Secondly, it is possible to use matrices of special types, for example, URNN [10] or Toeplitz matrices [9]. Finally, Tensor Train decomposition of layers can be used [11]. The variational dropout technique was originally used to adjust the individual dropout degree for each neuron [12]. In [13], the authors showed that variational dropout is able to eliminate certain neurons, thereby providing compression. If a prior distribution on individual weights is imposed, then we obtain sparse matrices. As in the case of ordinary pruning, this is not very convenient, therefor, in subsequent work, it is proposed to do structural compression, imposing a prior distribution immediately on the rows and columns of input matrices [14]. In this dissertation study, we analyzed in detail the current methods of compression of neural networks and proposed several new algorithms that solve this problem more effectively.

The goal of this dissertation study is the development of new algorithms of neural networks compression for natural language processing task. The proposed algorithms should solve the problem of compression more efficiently: they should use less resources (memory and processor power) of end devices, for example, mobile phones.

To achieve the goal, it is supposed to investigate various methods of compression of neural networks and to propose neural network architectures based on them that are of relatively small size and effectively solve applied problems. The main **tasks** of the study can be formulated:

1. An empirical analysis of neural network compression algorithms based on pruning and quantization techniques, matrix and tensor decompositions, and Bayesian methods.
2. Developing new and modifying existing neural networks compression algorithms, taking into account the peculiar properties of the natural language processing tasks and typical recurrent neural network architectures.
3. Adaptation of the developed algorithms to meet the specified characteristics for use on various devices such as mobile phones.

Main provisions for defense:

1. Development of matrix factorization based algorithms for compression of hidden layers of neural networks in the language modeling problem.

2. Effective and efficient solution of classification problem for large number of classes (10,000–30,000) based on matrix factorization techniques for the input and output layers of a given neural network
3. Adaptation of Bayesian dropout techniques to recurrent neural networks for the language modeling problem.
4. Efficient porting of recurrent neural networks to mobile devices and testing in laboratory settings.

The novelty:

1. This is the first time when TT-decomposition has been applied for recurrent neural networks in the language modeling problem. Moreover, the application of low-rank matrix factorization to the problem and the whole class of models based on recurrent neural networks have been investigated.
2. This is the first time when double stochastic variational inference algorithm has been applied to the recurrent neural network compression in the language modeling task.
3. Compressed models were ported to mobile devices. At that time (according to publications at leading conferences) it was the first implementation (documented in [15]) of neural networks (especially compressed) on a mobile GPU.
4. A general compression pipeline for recurrent neural networks for the language modeling problem have been proposed.

Theoretical and practical significance. In this work new methods for recurrent neural networks compression has been developed. These methods belong to different types: e.g., matrix-based and Bayesian. They take into account the high-dimensional input-output subproblem in the language modeling problem. An on-device implementation of these models to mobile GPUs has also been developed. It expands the scope of applications for recurrent neural networks and allows their more efficient exploitation on user devices.

Reliability of the results obtained is supported by mathematically correct compression models as well as by a large number of experiments with various architectures and data sets. The results are comparable with the state-of-the-art works of other authors.

Publications and probation of the thesis.

The main results of the thesis are published in [15–18].

First-tier publications

– Artem M. Grachev, Dmitry I. Ignatov and Andrey V. Savchenko. Neural Networks Compression for Language Modeling. — *Pattern Recognition and Machine Intelligence - 7th International Conference, PReMI 2017, Kolkata, India, December5-8, 2017, Proceedings. – 2017. – Pp. 351–35.*
– Elena Andreeva, Dmitry I. Ignatov, Artem M. Grachev and Andrey V. Savchenko. Extraction of Visual Features for Recommendation of Products via Deep Learning. *– Analysis of Images, Social Networks and Texts – 7th International Conference, AIST 2018, Moscow, Russia, July 5-7, 2018, Revised Selected Papers. – 2018. – Pp. 201–210.*
– Artem M. Grachev, Dmitry I. Ignatov and Andrey V. Savchenko. Compression of Recurrent Neural Networks for Efficient Language Modeling. — *Applied Soft Computing – 2019. – Vol. 79. – Pp. 354 – 362.*

Other publications

– Maxim Kodryan*, Artem Grachev*, Dmitry Ignatov and Dmitry Vetrov. Efficient Language Modeling with Automatic Relevance Determination in Recurrent Neural Networks — *ACL 2019, Proceedings of the 4th Workshop on Representation Learning for NLP, August 2, 2019, Florence, Italy*

Reports at conferences and seminars.

– 7th International Conference on Pattern Recognition and Machine Intelligence (06.12.2017, Kolkata, India). Topic: "Neural network compression for language modeling".
– PhD seminar at the Faculty of computer science, National Research University Higher School of Economics (2017, Moscow, Russia). Topic: "Neural network compression for language modeling".
– Open Systems publisher's conference on "Machine learning technology" (October 2018, Moscow, Russia). Topic: "Neural Networks for mobile devices".
– Seminar "Automatic text processing and analysis" at the National Research University Higher School of Economics (20.04.2019, Moscow, Russia). Topic "Methods of Recurrent Neural Network Compression for Natural Language Processing".

– Seminar at Samsung R&D Russia (2017, Moscow, Russia). Topic: "Methods of Recurrent Neural Network Compression for Natural Language Processing".
– Seminar at Samsung R&D Russia (2019, Moscow, Russia). Topic: "DSVI-ARD compression in neural networks".

The works [15–17] are recorded in the scientific citation system **Scopus**. The article [17] in the journal Applied Soft Computing [17] falls in Q1 by **Scopus** and Q1 by **Web of Science**. The work [18] was published in the workshop at the CORE A* conference and is included in **ACL Anthology**.

Personal contribution into the main provisions for defense. The results are personally obtained by the author, except the results obtained in collaboration with E. Andreeva in the paper [16] and M. Kodryan in the paper [18].

Contents of the work

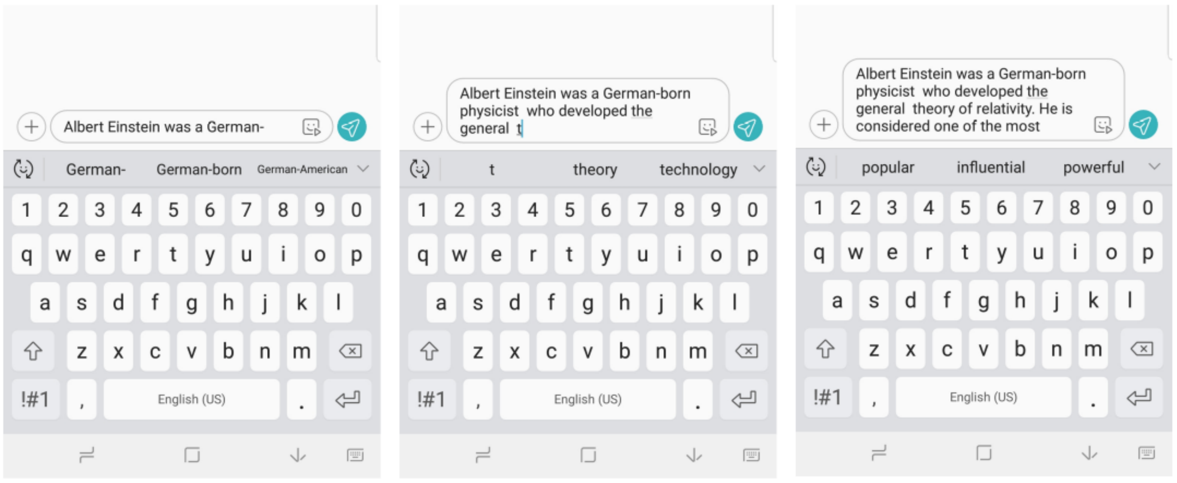
Volume and structure of the work. The thesis contains an introductory chapter, four main chapters, and a concluding chapter. The full volume of the thesis is 105 pages, including 17 figures and nine tables. The list of references contains 84 items.

In **introduction**, we discuss the relevance of the research studies presented in this thesis; the goals and tasks of the work are formulated; the scientific novelty of the work and the main provisions for defense are described.

The first chapter is introductory. Let us consider the language modeling problem, where it is required to estimate the probability of a sentence or sequence of words $(w_1, \dots, w_T)$ in a language L.

$$\mathsf{P}(w^1, \dots, w^T) = \mathsf{P}(w^1, \dots, w^{T-1})\mathsf{P}(w^T | w^1, \dots, w_{T-1}) = \\ = \prod_{t=1}^{T} \mathsf{P}(w^t | w^1, \dots, w^{t-1}) \quad (1)$$

This problem is relevant as a single one as well as in the context of various applications. For example, such language models are directly used in mobile keyboards (see Fig. 1).


Albert Einstein was a German-

Albert Einstein was a German-born physicist who developed the general t

Albert Einstein was a German-born physicist who developed the general theory of relativity. He is considered one of the most

Figure 1 — An example of the industrial application of the language modeling task: a mobile keyboard.

Further in this chapter various ways to solve this problem are discussed. For a long time, N-gram models [19; 20] and their various variations were widely used here. Then they were slowly replaced by recurrent neural networks [21; 22]. RNN's cell can be described as follows:

$$z_\ell^t = W_\ell x_{\ell-1}^t + U_\ell x_\ell^{t-1} + b_l \tag{2}$$

$$x_\ell^t = \tanh(z_\ell^t), \tag{3}$$

where $t \in \{1, \ldots, N\}$ is a current time step, $\ell \in \{1, \ldots, L\}$ is a current layer, W_ℓ and U_ℓ are the matrices of weights for layers ℓ, and tanh is a hyperbolic tangent, which is often used as an activation function in recurrent neural networks. A vector x_ℓ^t is an output vector for the layer ℓ at the time step t. We analyze such networks in detail. We consider their variations such as LSTM and GRU and also discuss the technique of backpropagation through time [23–25], which is used for training of such networks. We also discuss how the solution quality is measured in the context of this problem and various techniques that are used to improve convergence properties during the training phase.

In **the second chapter** we consider recurrent neural networks in terms of the used space on devices. Thus, all parameters of a neural network can be divided into two types. Parameters of the hidden layers and parameters of the input and ouput layers, the number of which closely related with the vocabulary size. We can count the total number of parameters as follows:

$$n_{total} = 8Lk^2 + 2|\mathbb{V}|k, \tag{4}$$

where L is the number of hidden layers, k is the size of hidden layers, $|\mathbb{V}|$ is the vocabulary size, and 8 is a constant corresponding to the number of matrices in the hidden layer (8 is the number of matrices for an LSTM cell, while for GRU it is six and two for RNN, respectively).

The remainder of this chapter provides an overview of the main approaches to compression of neural networks. There are two main directions: matrix decompositions based methods and methods based on matrix pruning. The chapter concludes with two basic approaches to compression: pruning and quantization.

Pruning is a method for reducing the number of parameters of a neural network by removing the weights that are nearly equal to zero. We can fix a threshold C and nullify all the weights that are less then C in absolute value, i.e. $|w_{ij}| < C$.

Quantization is a method for reducing the size of a compressed neural network in memory. In this technique, each floating-point value of a weight is packed into, e.g., 8-bit integer representing the closest real number in one of 256 equally-sized intervals within the whole range of the weight's values. Thus, if initially our number was stored in memory as a 32-bit float, now it is stored as an 8-bit integer and occupies four times less space.

Pruning and quantization have common disadvantages since they do not support training from scratch. Moreover, their practical usage is quite laborious. The reason for such behaviour of pruning mostly lies in the inefficiency of sparse computing. In the case of quantization, the model is stored in an 8-bit representation, but 32-bits computations are still required. It means that we do not obtain advantages using in-memory techniques at least until the special processors (like tensor processing unit, TPU) are not used. They are adapted for effective 8- and 16-bits computations.

The third chapter deals with compression methods that are related to matrix decompositions. Matrix multiplication operations are well optimized on modern processors[4] and the speed of their multiplication directly depends on the matrices size. That is, by reducing the size of the weights' matrices in the neural network, we simultaneously reduce the number of parameters and increase the speed of computations.

[4]Intel matrix multiplication

In this study, we applied low-rank matrix decomposition and Tensor Train decomposition [26] to recurrent neural network compression. For example, for a usual RNN cell, such a low-rank decomposition can be applied as follows:

$$x_l^t = \tanh\left[W_l^a m_{l-1}^t + U_l^a m_l^{t-1} + b_l\right] \tag{5}$$

$$m_l^t = U_l^b x_l^t \tag{6}$$

$$\tag{7}$$

Here $U_\ell^a, W_\ell^a \in R^{k\times r}$ and $U_\ell^b \in R^{r\times k}$, k are the original sizes of hidden layers, and r is a decomposition rank. The reduction of parameters is achieved due to the choice of the rank r under condition that $r \ll k$. Fig. 2 presents an illustrative example that shows how we apply matrix decomposition. In the thesis, we describe in details how this type of decomposition can be applied to various types of recurrent neural networks, such as LSTM and GRU.

Another decomposition we used here is the Tensor Train decomposition (TT-decomposition). It is one of the possible generalizations of SVD to tensors of dimension greater than two. The use of TT-decomposition is motivated by the possibility of converting matrices into their corresponding tensors, after which they are decomposed and compressed. Due to the fact that in the tensor we have several dimensions at once, we get a potentially higher compression rate and greater flexibility in compression.

Let us describe how we apply this decomposition for NN compression. Consider, for example, the weights matrix $W \in \mathbb{R}^{k\times k}$ of the RNN layer (2). One can arbitrarily choose the numbers $k_1, \ldots, k_d$ such that $k_1 \times \ldots \times k_d = k \times k$, and reshape the weights matrix to a tensor $\mathcal{W} \in \mathbb{R}^{k_1\times\ldots\times k_d}$. Here, d is the order (degree) of a tensor, $k_1, \ldots, k_d$ are the sizes of each its dimension. Thus, we can perform the TT-decomposition of the tensor $\mathcal{W}$ and obtain a set of matrices $G_m[i_m] \in \mathbb{R}^{r_{m-1}\times r_m}, i_m = 1, \ldots, k_m$, $m = 1, \ldots, d$ and $r_0 = r_d = 1$ such that each tensor element can be represented as $\mathcal{W}(i_1, i_2, \ldots, i_d) = G_1[i_1]G_2[i_2]\ldots G_d[i_d]$. Here, $r_0, \ldots r_m$ are the ranks of the decomposition. This TT-decomposition can be efficiently obtained with the TT-SVD algorithm described in [26]. In fact, each $G_m \in \mathbb{R}^{r_{m-1}\times k_m\times r_m}$ is a three-dimensional tensor with the second dimension k_m corresponding to the dimension of the original tensor and two ranks r_{m-1}, r_m, that, in a certain sense, represents the size of an internal representation for this dimension. It is necessary to emphasize that even with the fixed dimensions number of reshaped tensors and their sizes, we still have plenty of variants to choose the ranks in the TT-decomposition.

We denote both operations, converting the matrix W into the tensor $mathcalW$ and decomposition into the Tensor Train format, as a single operation $\mathrm{TT}(W)$. Applying it to both matrices W and V in the recurrent layer (2), we get the TT-RNN layer in the following form:

$$z_\ell^t = \tanh(\mathrm{TT}(W_l)x_{\ell-1}^t + \mathrm{TT}(U_l)x_\ell^{t-1} + b_\ell). \tag{8}$$

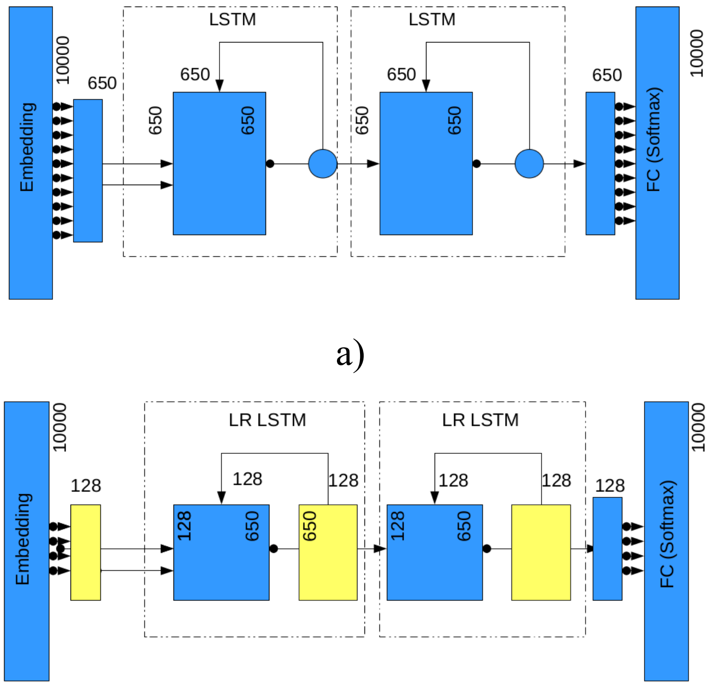


Figure 2 — Neural Network Architecture: (a) original LSTM 650-650 and (b) its low-rank model.

We conducted a series of experiments with different types of cells and their various sizes. It was shown that the main gain that we get from the compression of recurrent neural networks using low-rank matrix decompositions in comparison with pruning and quantization is almost no loss of computation speed. The gain in memory is almost directly transferred to the gain in speed. On the other hand, methods that work with sparse matrices result in larger compression in terms of memory used, but work longer in terms of computation time. Our experiments with models on mobile phone devices confirmed that low-rank compression of the **LR LSTM 650-650** model is the most effective in terms of both memory and computational complexity.

In **the fourth chapter**, we consider Bayesian methods for neural networks compression. In this context, Bayesian methods can be considered as mathematically better justified form of pruning. At the beginning of the chapter, we describe the general principles of Bayesian methods, then we consider the use of a variational dropout (VD) [27] for neural network pruning.

Next, we present a new method for compression of recurrent neural networks, which is based on automatic relevance determination with double stochastic variational inference (ARD DSVI). The main idea of this method lies in applying the method of searching for relevant features to the output layer of a neural network. Due to the fact that in our problem we usually deal with a large vocabulary, we get a high-dimensional problem. To solve it, we use stochastic approximation. In Eq. 9 the final functional is presented. Optimization of this functional leads to the optimal parameters of the posterior distribution $q\left(W \mid \boldsymbol{\mu}, \boldsymbol{\sigma}\right)$, and also the optimal λ^* for the prior distribution.

$$\begin{gathered}\mathcal{L}\left(q\left(W \mid \boldsymbol{\mu}, \boldsymbol{\sigma}\right), \Lambda^*\right) = \mathbb{E}_{W \sim q(W \mid \boldsymbol{\mu}, \boldsymbol{\sigma})}\left[\log p(Y \mid W, X)\right] + \\ + \frac{1}{2} \sum_{i=1}^{D} \sum_{j=1}^{K} \log \frac{\sigma_{ij}^2}{\mu_{ij}^2 + \sigma_{ij}^2} \longrightarrow \max_{\boldsymbol{\mu}, \boldsymbol{\sigma}}\end{gathered} \tag{9}$$

Choosing the clipping level λ_{thresh} on the validation set, we can choose the level at which we get the maximum compression rate with almost no loss of quality.

We adopted this method for pruning the output layer in our model. In addition, we first (by our best knowledge) considered the use of Bayesian pruning methods together with the use of weights sharing technique for the input-output layer. From the results of the experiments we can see that it is possible to achieve compression rate of the output layer up to 50 times in comparison with the original model (removal of 98% of the parameters). Certainly, this is even higher than using the matrix decomposition technique, but with the price of worse perplexity.

Bayesian compression methods allow us to achieve much bigger compression rate than, for example, matrix decomposition methods, but the question of the effective implementation of such methods for specific devices still remains open. At the end of the chapter some ideas towards research prospects and further development of these methods are formulated.

In the **conclusion** the main results of the work are summarized:

1. Matrix factorization based algorithms for compression of hidden layers of neural networks in the language modeling problem have been developed;
2. The classification problem for large number of classes (10,000–30,000) has been effectively and efficiently solved based on matrix factorization techniques for the input and output layers of a given neural network;
3. Bayesian compression techniques have been adopted to recurrent neural networks for the language modeling problem;
4. Efficient porting of recurrent neural networks to mobile devices and testing in laboratory setting have been done.

Bibliography

1. *Schmidhuber Jurgen*. Deep Learning in Neural Networks: An Overview // *Neural Networks*. — 2015. — Vol. 61. — Pp. 85–117.

2. *Bengio Yoshua*. Learning deep architectures for AI // *Foundations and Trends in Machine Learning*. — 2009. — Vol. 2, no. 1. — Pp. 1–127. — Also published as a book. Now Publishers, 2009.

3. *Deng Li, Yu Dong*. Deep Learning: Methods and Application // *Foundations and Trends in Signal Processing*. — 2013. — Vol. 7. — Pp. 197–387.

4. *Goodfellow Ian, Bengio Yoshua, Courville Aaron*. Deep Learning. — MIT Press, 2016. — http://www.deeplearningbook.org.

5. *Cybenko George*. Approximation by superpositions of a sigmoidal function // *MCSS*. — 1989. — Vol. 2, no. 4. — Pp. 303–314. https://doi.org/10.1007/BF02551274.

6. *Hornik Kurt, Stinchcombe Maxwell B., White Halbert*. Multilayer feedforward networks are universal approximators // *Neural Networks*. — 1989. — Vol. 2, no. 5. — Pp. 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

7. Learning both Weights and Connections for Efficient Neural Network / Song Han, Jeff Pool, John Tran, William J. Dally // Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. — 2015. — Pp. 1135–1143. http://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-ne

8. *Han Song, Mao Huizi, Dally William J.* Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding // *CoRR*. — 2015. — Vol. abs/1510.00149. http://arxiv.org/abs/1510.00149.

9. *Lu Zhiyun, Sindhwani Vikas, Sainath Tara N.* Learning compact recurrent neural networks // 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016. — 2016. — Pp. 5960–5964. https://doi.org/10.1109/ICASSP.2016.7472821.

10. *Arjovsky Martín, Shah Amar, Bengio Yoshua*. Unitary Evolution Recurrent Neural Networks // Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016. — 2016. — Pp. 1120–1128. http://jmlr.org/proceedings/papers/v48/arjovsky16.html.

11. Tensorizing Neural Networks / Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, Dmitry P. Vetrov // Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. — 2015. — Pp. 442–450. http://papers.nips.cc/paper/5787-tensorizing-neural-networks.

12. *Kingma Diederik P, Salimans Tim, Welling Max*. Variational Dropout and the Local Reparameterization Trick // Advances in Neural Information Processing Systems 28 / Ed. by C. Cortes, N. D. Lawrence, D. D. Lee et al. — Curran Associates, Inc., 2015. — Pp. 2575–2583. http://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-t
pdf.

13. *Molchanov Dmitry, Ashukha Arsenii, Vetrov Dmitry P.* Variational Dropout Sparsifies Deep Neural Networks // Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. — 2017. — Pp. 2498–2507. http://proceedings.mlr.press/v70/molchanov17a.html.

14. Structured Bayesian Pruning via Log-Normal Multiplicative Noise / Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, Dmitry P. Vetrov // Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. — 2017. — Pp. 6778–6787. http://papers.nips.cc/paper/7254-structured-bayesian-pruning-via-log-normal-multiplicat

15. *Grachev Artem M., Ignatov Dmitry I., Savchenko Andrey V.* Neural Networks Compression for Language Modeling // Pattern Recognition and Machine Intelligence - 7th International Conference, PReMI 2017, Kolkata, India, December 5-8, 2017, Proceedings. — 2017. — Pp. 351–357. https://doi.org/10.1007/978-3-319-69900-4_44.

16. Extraction of Visual Features for Recommendation of Products via Deep Learning / Elena Andreeva, Dmitry I. Ignatov, Artem M. Grachev, Andrey V. Savchenko // Analysis of Images, Social Networks and Texts - 7th International Conference, AIST 2018, Moscow, Russia, July 5-7, 2018, Revised Selected Papers. — 2018. — Pp. 201–210. https://doi.org/10.1007/978-3-030-11027-7_20.

17. *Grachev Artem M., Ignatov Dmitry I., Savchenko Andrey V.* Compression of recurrent neural networks for efficient language modeling // *Applied Soft Computing*. — 2019. — Vol. 79. — Pp. 354 – 362. http://www.sciencedirect.com/science/article/pii/S1568494619301851.

18. Efficient Language Modeling with Automatic Relevance Determination in Recurrent Neural Networks / Maxim Kodryan, Artem Grachev, Dmitry Ignatov, Dmitry Vetrov // ACL 2019, Proceedings of the 4th Workshop on Representation Learning for NLP, August 2, 2019, Florence, Italy. — 2019.

19. *Kneser Reinhard, Ney Hermann.* Improved backing-off for M-gram language modeling // 1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995. — 1995. — Pp. 181–184. https://doi.org/10.1109/ICASSP.1995.479394.

20. *Jelinek Frederick.* Statistical Methods for Speech Recognition. — MIT Press, 1997. https://mitpress.mit.edu/books/statistical-methods-speech-recognition.

21. A Neural Probabilistic Language Model / Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Janvin // *Journal of Machine Learning Research.* — 2003. — Vol. 3. — Pp. 1137–1155. http://www.jmlr.org/papers/v3/bengio03a.html.

22. *Mikolov Tomáš.* Statistical Language Models Based on Neural Networks: Ph.D. thesis / Brno University of Technology. — 2012. — P. 129. http://www.fit.vutbr.cz/research/view_pub.php?id=10158.

23. *Werbos P.* Backpropagation through time: what it does and how to do it // *Proceedings of the IEEE.* — 1990. — Vol. 78(10). — Pp. 1550–1560.

24. *Rumelhart David E., Hinton Geoffrey E., Williams Ronald J.* Neurocomputing: Foundations of Research / Ed. by James A. Anderson, Edward Rosenfeld. — Cambridge, MA, USA: MIT Press, 1988. — Pp. 696–699. http://dl.acm.org/citation.cfm?id=65669.104451.

25. *Pascanu Razvan, Mikolov Tomas, Bengio Yoshua.* On the difficulty of training recurrent neural networks // Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013. — 2013. — Pp. 1310–1318. http://jmlr.org/proceedings/papers/v28/pascanu13.html.

26. *Oseledets Ivan V.* Tensor-Train Decomposition // *SIAM J. Scientific Computing.* — 2011. — Vol. 33, no. 5. — Pp. 2295–2317. http://dx.doi.org/10.1137/090752286.

27. *Chirkova Nadezhda, Lobacheva Ekaterina, Vetrov Dmitry P.* Bayesian Compression for Natural Language Processing // Proceedings of the 2018 Conference

on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018. — 2018. — Pp. 2910–2915. https://aclanthology.info/papers/D18-1319/d18-1319.