A Survey of Deep Learning Methods and Software Tools for Image Classification and Object Detection[1,2]

P. N. Druzhkov and V. D. Kustikova

Lobachevsky State University of Nizhni Novgorod, Institute of Information Technologies, Mathematics and Mechanics, Nizhni Novgorod, Russian Federation

e-mail: itlab.ml@cs.vmk.unn.ru

Abstract—Deep learning methods for image classification and object detection are overviewed. In particular we consider such deep models as autoencoders, restricted Boltzmann machines and convolutional neural networks. Existing software packages for deep learning problems are compared.

Keywords: deep learning; image classification; object detection; sparse coding; autoencoder; restricted Boltzmann machine; convolutional neural networks.

DOI: 10.1134/S1054661816010065

INTRODUCTION

Since the first works on artificial neural networks they have experienced lots of ups and downs, but have always been of special interest for researchers. Neural networks-based methods have been successfully applied to classification, clustering, forecasting, approximation and recognition problems in medicine, biology, commerce, robotics etc. The latest advance in the field has been caused by the invention of deep learning methods [1–3], induced by the progress of parallel computing hardware and software. The key component of deep learning is the multilayered hierarchical data representation typically in the form of a neural network with more than two layers. Such methods allow automatically synthesizing data descriptions (features) of a higher level based on the lower ones. In terms of image analysis hierarchy levels can correspond to "pixels → edges → combinations of edges" chain. Though deep learning has been expired by neural networks there are some attempts to apply its ideas to other types of models.

Here a survey of deep learning methods aimed at image classification and object detection in images is represented. The applicability of the methods under consideration to these problems is confirmed by the latest results of well-known competitions such as ImageNet [5] and PASCAL Visual Object Challenge [6], in the context of which the breakthrough in image classification task has been recently made [15, 64].

[1] This paper uses the materials of the report submitted at the 9th Open German-Russian Workshop on Pattern Recognition and Image Understanding, held in Koblenz, December 1–5, 2014 (OGRW-9-2014).

[2] The article published in the original.

Received August 06, 2015

IMAGE CLASSIFICATION PROBLEM

Image classification problem requires determining the category (class) that it belongs to. The problem is considerably complicated with the growth of categories' count, if several objects of different classes are present at the image and if the semantic classes' hierarchy is of interest, because image can belong to several categories simultaneously. Fuzzy classes place another difficulty of probabilistic categories' assignment.

IMAGE CLASSIFICATION METHODS

Sparse Coding

Bag-of-words methods [7] have been among the most popular and successive approaches for solving classification problems before the expansion of deep learning. One of the most advanced methods of this type is *sparse coding* [8]. It maps the initial image description $x \in \Re^d$ (with reference to image classification it can be a vector of pixels' intensities, dense SIFT-descriptor [9, 10] etc.) to a vector $x \in \Re^m$ with lots of zero components, such that $x \approx D \cdot s$, where $D \in \Re^{d \times m}$ and m can be much greater than d. D is called a *dictionary* and s is a *code*. With a fixed dictionary the problem of new representation (i.e. code) computing is stated as $s^* = \arg\min_s \left(Err\left(x, D \cdot s\right) + \lambda \Psi(s) \right)$, where $Err\left(x, D \cdot s\right)$ component is responsible for the coding error, and, generally equals to $\left(\left\| x - D \cdot s \right\|_2 \right)^2$ and $\Psi(s)$ defines a code sparseness constraint, for example, as $\|s\|_0$ or $\|s\|_1$. Dictionary can be learned from data X, solving $D^* = \arg\min_D \sum_{x \in X} \min_s \left(Err\left(x, D \cdot s\right) + \lambda \Psi\left(s\right) \right)$.

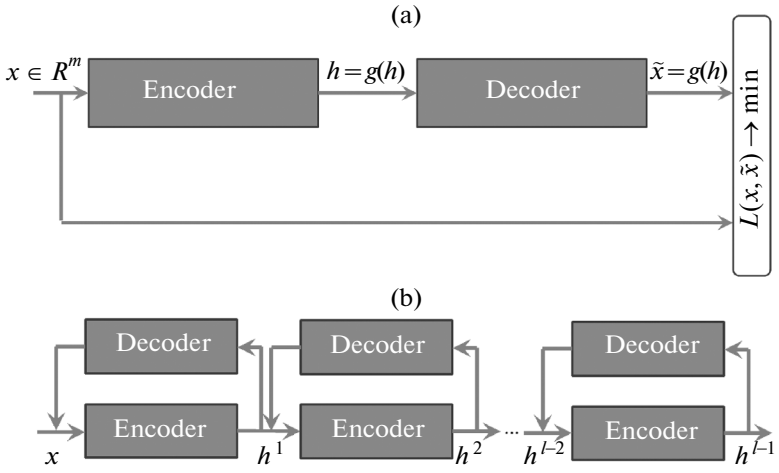

Fig. 1. Scheme of autoencoder and stacked autoencoder.

The common sparse coding classification pipeline is as follows. The dictionary is learned from a set of unlabeled images, code is computed for each labeled image at hand, and a classifier is trained trying to predict the class by the code. As labeled data is not required at the dictionary learning stage, this approach is advantageous in scarce labeled data situations. In opposite case dictionary learning can be considered in a supervised manner, providing additional information to improve features' quality [11–13]. Sparse coding as it described here is not capable of building features' hierarchies and it is not straightforward to simply stack one coding model on top of the other [63]. Though rather successful attempts to make sparse coding deep exist [10, 14] but there is still room for improvement. It should be also mentioned that sparse coding is not the only algorithm that has been attempted to make it deep [65, 66].

Deep Learning Models: Autoencoders, Restricted Boltzmann Machines, Convolutional Neural Networks

During the past years deep learning approaches that are heavily based on neural networks have been of special interest. Each node of a network (i.e. artificial neuron) is associated with a feature, and neurons of the subsequent layer generalize essential features from the previous one. The big amount of trainable parameters leads to necessity of network topology and activation functions constraining and development of highly parallel algorithms for training. Therefore considerable emphasis is placed on searching of networks' topologies that better suit for image classification and effective methods for their training. One of the common training techniques is to use the unsupervised pre-training stage that allows a preliminary fitting using only unlabeled data. It is proved to be a good starting point for a subsequent supervised fine-tuning. In this context two main types of models can be distinguished:

Autoencoder (AE) [20]. AE performs coding with loss of information so that the result of subsequent decoding is as close to the original data as possible (Fig. 1a). In general coding function can be represented as $h = f(x) = s_f(Wx + b_h)$, and decoding one as $y = g(h) = s_g(W^T h + b_y)$. Code h is nothing more than a feature vector of the current level of hierarchy. AE is designed to find such values of parameters $\{W, b_h, b_y\}$ that lead to minimal deviation of y from x, defined by the loss function. Different optimization techniques for parameters fitting exist and the most popular one is a *back propagation* (BP) method. Depending on a nonlinear coding function part s_f and a loss type AEs are subdivided into *sparse* (SAE) [21], *contractive* (CAE) [22] and *denoising* (DAE) [23]. Using an AE to some extent allows filtering out insignificant details for visual object modeling. AEs can be stacked (StAE) (Fig. 1b) to produce the hierarchy of features [23].

Restricted Boltzmann machine (RBM) [25, 33, 62]. As opposed to AEs RBMs are stochastic neural models. An RBM is a neural network with two layers corresponding to visible and hidden states of a probabilistic system. Each node of one layer is connected to every node of the other layer (Fig. 2). Visible neurons correspond to a given initial feature description and hidden ones—to features derived as functions of visible variables. RBM defines the probability distribution on the set of its visible states and training objective is to maximize the likelihood of a training sample. Effective methods for RBM training have been designed. Among them are *contrastive divergence* (CD) and its k-step (kCD) and persistent (PCD) [33] variations. RBMs can be used in deep models such as *deep belief networks* (DBN) [26], *deep Boltzmann machines*

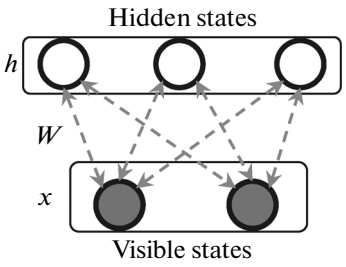

Fig. 2. Restricted Boltzmann Machine.

(DBM) [34] and be applied for deep AEs [35] and convolutional neural networks [26] pre-training.

Pre-training stage can be avoided if using *convolutional neural networks* (CNNs) [36]. Each layer of CNN represents a *feature map*. Feature map of an input layer is a matrix of pixel intensities (a separate matrix for each color channel). And feature map of any internal layer is an induced multi-channel image, where every "pixel" corresponds to a specific feature. Every neuron is connected with a small portion of adjacent neurons from the previous layer (receptive field). It is typical to interleave the layers doing different types of transformations [15, 24, 37, 38] on feature maps, such as filtering and pooling (Fig. 3). *Filtering* function computes a discrete convolution of filter-matrix with a receptive field neurons' values followed by a non-linear transform such as sigmoid and *pooling* is a possibly non-linear transformation that allows summarizing a receptive filed by one value making feature descriptions more robust. Max, average, L_2-polling are among the most popular choices. *Local contrast normalization* [41] is another operation that has proved to be useful in CNNs.

As soon as initial features hierarchy is constructed it can be fine-tuned in a supervised manner [4]. Final layer is added to the network such that each output neuron gives a conditional probability that the input image belongs to a specific class. Sigmoid and softmax are typically used as activation functions for this layer [15] and *mean squared error* $MSE = \frac{1}{n}\sum_{1\le i\le n}(y_i - y_i^*)^2$ or *cross-entropy loss* $L = -\sum_{1\le i\le n}\log(P(y_i = y_i^*))$ is minimized via the *stochastic gradient descent* (SGD) method. If fine-tuning of features is not required any type of classifier such as SVM [16] can be simply applied to the output of the final layer.

Resulting classifier can be applied to an arbitrary image of the same size, as those from the training set. If the image has a different resolution it can be scaled and cropped as in [15].

OBJECT DETECTION METHODS

Detection problem is more general in sense that it requires not only to determine whether the object of interest is present in image but also tell where all of its instances are located. Object detection is still a challenging problem due to a big amount of factors that must be handled: variety of possible objects' forms and colors, occlusions, lighting conditions, perspective etc.

Detection methods can be distinguished into three groups [27]: approaches, based on features extraction [28, 29]; template searching methods [30, 31]; movement detection [32]. Deep learning methods fall into the first group. The most straightforward way is to apply a deep classifier, trained as considered above, to regions of interest generated by the extensive sliding window approach [18, 17, 37], or some more cost efficient method [19, 39, 40]. Nevertheless determination on object position, size and scale can be embedded into the neural network [17, 18, 25] by adding layers of a special type.

Experimental results have shown that deep detectors are among the best-performing modern models, but no extreme improvement of the state-of-the-art has been performed yet.

DEEP LEARNING SOFTWARE TOOLS

Software tools implementing deep learning methods include libraries and packages [44–49, 53, 55, 56, 58–61], programming language extensions [52, 57] and self-contained languages [54]. Provided functionality varies a lot (table) raising the problem of choosing an appropriate tool.

A great batch of tools support a broad series of models including fully connected neural networks (FCNNs), CNNs, AEs and RBMs and implement popular training methods and loss functions. Deep Learn Toolbox [45], Theano [53], Pylearn2 [55], Deepnet [56] and DeepMat [49] are among them. Darch [50] package for R [51] also falls into this category, but it doesn't support CNNs. It is worth men-

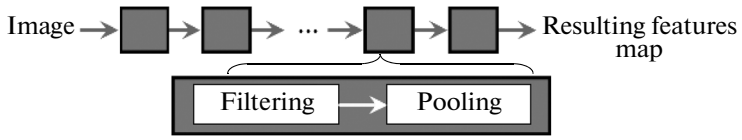

Fig. 3. Structure of a convolutional neural network.

Table 1. Comparison of deep learning software tools

Name	Language	OS	FC NNs	CNNs	AE	RBMs	Learning method	Loss
DeepLearn-Toolbox [45]	Matlab	Win, Lin	+ (Drop-out)	+	+ (StAE, DAE)	+ (DBN)	BP	MSE
Theano [53]	Python	Win, Lin, Mac	+	+	+ (DAE)	+ (DBN)	SGD	1. Negative Log-Likelihood 2. Zero-One
Pylearn2[1] [55]	Python	Lin, Vagrand	+ (Maxout Networks)	+	+ (StAE, CAE, DAE)	+ (GRBM[2], ssRBM[3])	SGD, BGD[4], NCG[5] (line search)[6]	1. Cross-entropy 2. Log-Likelihood
Deepnet[7] [56]	Python	Lin	+ (Drop-out)	+	+	+ (DBN, DBM)	GD, LBFGS[8], CD, PCD	1. Squared 2. Cross-entropy 3. Hinge
Deepmat [49]	Matlab	?	+ (Drop-out)	+	+ (DAE, StDAE[9])	+ (Gaussian RBM, DBN, DBM)	SBP[10], Ada-grad, CD, PCD	?
Darch [50]	R	Win, Lin	+	–	+	+ (DBN)	BP, CG[11], CD	1. MSE 2. Cross-entropy 3. Quadratic error
Torch [52]	Lua, C	Lin, iOS, Android	+	+	+	–	SGD, LBFGS, CG	1. MSE 2. Binary cross-entropy 3. Hinge 4. Others
Caffe [43]	C++, Python, Matlab	Lin	+ (Drop-out)	+	–[12]	–	SGD	1. Euclidean 2. Hinge 3. Info gain 4. Logistic 5. Sigmoid cross-entropy 6. Softmax
nnForge [44]	C++	Lin	+ (Dropout, MaxoutNet works)	+	–	–	SGD, SDLMA[13]	1. MSE 2. Squared Hinge 3. Negative Log-Likelihood 4. Cross-Entropy
CXXNET [60]	C++	Lin	+ (Dropout, Drop Con-nection)	+	–	–	SGD	?
Cudaconv-net [46]	C++	Win, Lin	+	+	–	–	?	1. Logistic regres-sion 2. Sum-of-squares
Cuda CNN [48]	Matlab	Win, Lin	+	+	–	–	SGD, SDLMA	?
EBLearn [47]	C++	Lin, Mac	+	+	–	–	BP	Energy-based functions

[1] Based on Theano, uses Cuda-convnet, implements differentiable sparse coding and spike-and-slab sparse coding.
[2] Gaussian RBM (GRBM).
[3] The spike-and-slab RBM (ssRBM).
[4] Batch Gradient Descent (BGD).
[5] Nonlinear conjugate gradient descent (NCG).
[6] BGD and NCG with searching for an optimal local solution at every step.
[7] Uses cudamat and cuda-convnet.
[8] Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm.
[9] Stacked Denoising Auto-Encoders (StDAE).
[10] Stochastic Back Propagation (SBP).
[11] Conjugate gradient method (CG).
[12] Autoencoder may be constructed.
[13] Stochastic Diagonal Levenberg-Marquardt algorithm (SDLMA).

Table 1. Contd.

Name	Language	OS	FC NNs	CNNs	AE	RBMs	Learning method	Loss
Hebel [59]	Python	Win, Lin	+ (Dropout)	in development	in development	in development	SGD	?
Crino[14] [61]	Python	?	+	– through [53]	+	– through [53]	SGD, BP	1. MSE 2. Cross-entropy 3. Mean Absolute Error
Lush [54]	Lush	Win (Cygwin), Lin	+	+	–	–	SGD, BGD, SLMA, BCG[15]	1. MSE 2. ?
R-CNN[16] [42]	Matlab	Lin	– through [43]	+	–	–	– through [43]	– through [43]

[14] Based on Theano, implements MLP, DNN, IODA.
[15] Batch Conjugate Gradient (BCG).
[16] Based on Caffe and provides an interface to it, implements a special type of CNNs [19].

tioning that Deep Learn Toolbox is solely implemented in MATLAB. It leads to some performance issues and forces to be cautious with the use of big data. Torch [52], Caffe [43], nnForge [44], CXXNET [60], Cuda-convnet [46] and Cuda CNN [48] represent the group of deep learning tools aimed at high-performance training of CNNs using GPUs through CUDA. Some of them are used inside the tools mentioned above. The rest of tools mostly implement varieties of deep neural networks using other libraries. For example, Crino [61] and R-CNN [42] are based on Theano and Caffe respectively. Lush [54] programming language is also an interface to Torch library.

This overview doesn't cover all of the available tools because of their vast and steadily increasing amount [67, 68].

ACKNOWLEDGMENTS

This work is supported by Russian Foundation for Basic Research (project No. 14-07-31269) and has been done in the "Information Technology" laboratory at Institute of Information Technologies, Mathematics and Mechanics of Lobachevsky State University of Nizhni Novgorod.

REFERENCES

1. G. E. Hinton, "Learning multiple layers of representation," Trends Cognitive Sci. **11**, 428–434 (2007).
2. J. Schmidhuber, Deep learning in neural networks: an overview. http://arxiv.org/abs/1404.7828
3. Resources and pointers to information about Deep Learning. http://deeplearning.net
4. D. P. Vetrov, "Machine learning: current state and perspectives," in *Proc. of RCDL* (Yaroslavl, 2013), Vol. 1, pp. 21–28.
5. ImageNet. http://www.image-net.org
6. PASCAL Visual Object Challenge. http://pascallin.ecs.soton.ac.uk/challenges/VOC
7. C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka, "Visual categorization with bags of keypoints," in *Proc. ECCV Int. Workshop on Statistical Learning in CV* (Prague, 2004).
8. H. Lee, A. Battle, R. Raina, and A.Y. Ng, "Efficient sparse coding algorithms," in *Proc. of NIPS* (Vancouver, 2006), pp. 801–808.
9. D. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vision **60** (2), 91–110 (2004).
10. Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi, "Unsupervised feature learning by deep sparse coding," in *Proc. of SIAM Int. Conf. on Data Mining* (Philadelphia, 2014), pp. 902–910.
11. J. Yang, K. Yu, and T. Huang, "Supervised translation-invariant sparse coding," in *Proc. of CVPR* (San Francisco, 2010), pp. 3517–3524.
12. Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," in *Proc. of CVPR* (San Francisco, 2010), pp. 2691–2698.
13. Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," in *Proc. of CVPR* (Colorado Springs, 2011), pp. 1697–1704.
14. A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. of Int. Conf. on Artificial Intelligence and Statistics* (Ft. Lauderdale, FL, 2011), Vol. 15, pp. 215–223.
15. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. of NIPS* (Lake Tahoe, 2012), pp. 1097–1105.
16. K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep fisher networks for large-scale image classification," in *Proc. of NIPS* (Lake Tahoe, 2013), pp. 163–171.
17. C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Proc. of NIPS* (Lake Tahoe, 2013), pp. 2553–2561.

18. D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proc. of CVPR* (Columbus, OH, 2014).
19. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. of CVPR* (Columbus, OH, 2014), pp. 580–587.
20. M. Hayat, M. Bennamoun, and S. An, "Learning nonlinear reconstruction models for image set classification," in *Proc. of CVPR* (Columbus, OH, 2014).
21. M. Ranzato, C. Poultney, and S. Chopra, "Efficient learning of sparse representations with an energy-based model," in *Proc. of NIPS* (Vancouver, 2006), pp. 1137–1144.
22. S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: explicit invariance during feature extraction," in *Proc. of ICML* (Bellevue, 2011), pp. 833–840.
23. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," J. Mach. Learn. Res. **11**, 3371–3408 (2010).
24. K. Kavukcuoglu, P. Sermanet, Y-lan Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *Proc. of NIPS* (Vancouver, 2010), pp. 1090–1098.
25. P. Luo, Y. Tian, X. Wang, and X. Tan, "Switchable deep network for pedestrian detection," in *Proc. of CVPR* (Columbus, OH, 2014).
26. H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. of ICML* (Montreal, 2009), pp. 609–616.
27. V. D. Kustikova, N. Yu. Zolotykh, and I. B. Meyerov, "A review of vehicle detection and tracking methods in video," Vestn. Lobachevsky State Univ. Nizhni Novgorod, No. 5(2), 347–357 (2012).
28. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," IEEE Trans. PAMI'10 **32** (9), 1627–1645 (2010).
29. J. Shotton, A. Blake, and R. Cipolla, "Contour-based learning for object detection," in *Proc. ICCV* (Beijing, 2005), Vol. 1, pp. 503–510.
30. C. H. Hilario, J. M. Collado, J. M. Armingol, and A. de la Escalera, "Pyramidal image analysis for vehicle detection," in *Proc. of Intelligent Vehicles Symp.* (Las Vegas, 2005), pp. 88–93.
31. Y. Amit, *2D Object Detection and Recognition: Models, Algorithms and Networks* (MIT Press, 2002).
32. M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis and Machine Vision* (Thomson, 2008).
33. Restricted Boltzmann Machines (RBMs). http://www.deeplearning.net/tutorial/rbm.html. Assessed 07.08.2014.
34. R. Salakhutdinov and G. Hinton, Deep Boltzmann Machines, DBMs. http://www.cs.toronto.edu/~fritz/absps/dbm.pdf
35. Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng, "Building high-level features using large scale unsupervised learning," in *Proc. of ICML* (Edinburgh, 2012).
36. Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. of ISCAS* (Paris, 2010), pp. 253–256.
37. M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Weakly supervised object recognition with convolutional neural networks," in *Proc. of NIPS* (Montreal, 2014).
38. M. Oquab, L. Bottou, I. Laptev, and J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks (2013). http://hal.inria.fr/docs/00/91/11/79/PDF/paper.pdf
39. J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," Int. J. Comput. Vision **104** (2), 154–171 (2013).
40. X. Wang, M. Yang, S. Zhu, and Y. Lin, "Regionlets for generic object detection," in *Proc. of ICCV* (Sydney, 2013).
41. K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun, "Learning invariant features through topographic filter maps," in *Proc. of CVPR* (Miami, 2009), pp. 1605–1612.
42. R-CNN – a visual object detection system. https://github.com/rbgirshick/rcnn
43. Caffe – a deep learning framework. http://caffe.berkeleyvision.org
44. nnForgeLibrary. http://milakov.github.io/nnForge
45. DeapLearnToolbox. https://github.com/rasmusbergpalm/DeepLearnToolbox
46. Cuda-convnet – high-performance C++/CUDA implementation of convolutional neural networks. http://code.google.com/p/cuda-convnet
47. EBLearn – a machine learning library. http://eblearn.sourceforge.net
48. Cuda CNN Library. http://www.mathworks.com/matlabcentral/fileexchange/24291-cnnconvolutional-neural-network-class , https://bitbucket.org/intelligenceagent/cudacnnpublic/wiki/Home
49. DeepMat Library. https://github.com/kyunghyuncho/deepmat
50. Package Darch. http://cran.r-project.org/web/packages/darch/index.html
51. Software Environment R. http://www.r-project.org
52. Torch – a scientific computing framework. http://www.torch.ch
53. Theano Library. https://github.com/Theano/Theano , http://deeplearning.net/software/theano
54. Lush programming language. http://lush.sourceforge.net
55. Pylearn2 – a machine learning library. http://deeplearning.net/software/pylearn2
56. Deepnet Library. https://github.com/nitishsrivastava/deepnet
57. DeCAFFramework. https://github.com/UCB-ICSI-Vision-Group/decaf-release
58. Cuda-convnet NYU. http://cs.nyu.edu/~wanli/dropc
59. Hebel – GPU-accelerated deep learning library. https://github.com/hannes-brt/hebel
60. CXXNET – a neural network toolkit. https://github.com/antinucleon/cxxnet

61. Crino — a neural network library. https://github.com/jlerouge/crino

62. A. Courville, J. Bergstra, and Y. Bengio, A spike and slab restricted Boltzmann machine (2011). http://jmlr.org/proceedings/papers/v31/luo13a.pdf

63. Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi, "Unsupervised feature learning by deep sparse coding," in *Proc. of ICDM* (Shenzhen, 2014), pp. 902–910.

64. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, ImageNet large scale visual recognition challenge. http://arxiv.org/abs/1409.0575

65. C. Vens and F. Costa, "Random forest based feature induction," in *Proc. of ICDM* (Vancouver, 2011), pp. 744–753.

66. V. Yu. Martyanov, A. N. Polovinkin, and E. V. Tuv, " Image classification with codebook based on decision tree ensembles," in *Proc. of Intelligent Information Processing Conf.* (Guilin, 2012), pp. 480–482.

67. The Intel® Deep Learning Framework (IDLF). https://github.com/01org/idlf

68. Scikit-neuralnetwork Library. https://github.com/aigamedev/scikit-neuralnetwork

Pavel Nikolaevich Druzhkov

Born 1989 Graduated Lobachevsky State University of Nizni Novgorod in 2012. He is a junior research of the Lobachevsky State University of Nizhni Novgorod.

Research interests: machine learning and data mining, computer vision.

Number of publications (monographs and articles): 6.

Valentina Dmitrievna Kustikova

Born 1987. Graduated in 2010, Lobachevsky State University of Nizhni Novgorod. Year of dissertation completion (Candidate's, Doctoral): 2015, Candidate of Engineering Sciences.

Assistent at the Lobachevsky State University of Nizhni Novgorod.

Research interests: computer vision, machine learning, parallel computing.

Number of publications (monographs and articles): 8.