Deep Learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which One Is the Best in Speed and Accuracy?

Conference Paper · October 2016

READS

720

3 authors, including:

Vassili Kovalev

National Academy of Sciences of Belarus

79 PUBLICATIONS **644** CITATIONS

SEE PROFILE

Alexander Kalinovsky

United Institute of Informatics Problems

8 PUBLICATIONS **1** CITATION

SEE PROFILE

Deep Learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which One Is the Best in Speed and Accuracy?

Vassili Kovalev [1], Alexander Kalinovsky [1], Sergey Kovalev [2]
[1] United Institute of Informatics Problems, Belarus National Academy of Sciences
Surganova St., 6, 220012 Minsk, Belarus
vassili.kovalev@gmail.com, gakarak@gmail.com, http://imlab.grid.by/

[2] Altoros, 9 Dombrovskaya St., Fl 5, Minsk, Belarus 220140
sergey.kovalev@altoros.com, www.altoros.com

Abstract: *This paper presents the results of a comparative study of the leading Deep Learning frameworks, including Theano (with a Keras wrapper), Torch, Caffe, TensorFlow, and Deeplearning4J. Detailed results of quantitative assessment of their training and predicting speed, as well as resultant classification accuracy, are provided. The research was conducted jointly by the United Institute of Informatics Problems (Belarus National Academy of Sciences) and Altoros, a global provider of big data and Platform-as-a-Service solutions.*

Keywords: Deep Learning, Fully Connected Neural Network, Convolutional Neural Networks, Comparison.

1. INTRODUCTION

Fully Connected Neural Network (FCNN). In general, the FCNN networks can be viewed as Feedforward Multilayer Perceptrons. The "feed-forward" term stresses the fact that connections between the units do not form a cycle. This is opposed to the Recurrent Neural Networks where presence of cycles allows introducing into the network certain elements of dynamic behavior that is dependence of the output results on "time".

The main purpose of FCNN is data classification. As stated by the Universal Approximation Theorem [2], any continuous function (i.e., classifier) can be represented by feed-forward network (i.e., by a multilayer perceptron), with a single hidden layer containing a finite number of neurons. The only problem remains that the theorem does not impose upper limit for the size of the single intermediate layer. Therefore, in practice researchers vary the number of intermediate layers instead.

Recently, there can be an explosion of interests observed to the new methods and software solutions, which capitalize on so-called Deep Learning approaches, methods, and "deep" neural networks of new generation. Such a great interest to the deep neural networks can be partly explained by the fact that since 2009, they have won many official international pattern recognition competitions, achieving the first superhuman visual pattern recognition results in limited domains (see [3] for up-to-date review of the field). Lately, several software frameworks have been developed, which implement different Deep Learning methods. Typically, these frameworks distributed freely and can be used as a core for building up specific software solutions in a wide range of practical scenario. However, it is not obvious which one is best suited for every particular case. The problem of selection is complicated by a number of factors, one of the main of which is the lack of objective quantitative characteristics of available Deep Learning frameworks.

Thus, the purpose of this study was to examine the leading Deep Learning frameworks including Theano with Keras wrapper [4], Torch [5], Caffe [6], TensorFlow [7], and Deeplearning4J [8] for quantitative assessment of such key characteristics as their speed and classification accuracy. Some other important features are also highlighted. In all the occasions, we limit ourselves here by Fully Connected networks only. It is expected that this work would start a small series of publications, which will cover other network architectures and application issues as well.

Finally, it should be noted that similar to other comparative studies, results are mostly described here not as a plain text but rather presented in form of tables and plots what ease understanding of reported materials for potential readers.

2. INPUT TEST DATA

A MNIST (Mixed National Institute of Standards and Technology) database of handwritten digits [9] was used for testing the five leading Deep Learning FCNN frameworks. It is a large database of handwritten digits, which is commonly used for training various image classification approaches. Nowadays, the database is also widely used for training and testing in the field of Machine Learning. It was created by "re-mixing" the samples from NIST's original datasets. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels. Examples of digit images are shown in Fig. 1.

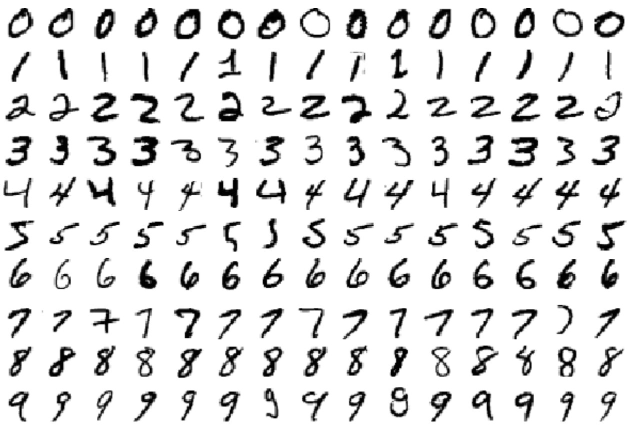
Fig.1 – Examples of MNIST digit images.

The digit images were inputted to the tested neural networks pixel-wise as vectors of 784 elements whose values ranged from 0 to 255.

The purpose of classification task used for testing was to categorize image of every digit to its "true" class.

A total of 60 000 digit images were used as a training set and 10 000 other constituted the test set, which used at the digit prediction (categorization) stage.

3. METHODS

As the title of the paper suggests, the main subjects of this experimental comparative study was the speed of convergence of deep neural networks of FCNN type at the training stage as well as the classification accuracy of trained networks at the prediction stage. Specifically, the quantitative metrics employed include:

(a) Convergence time.

(b) Prediction (classification) time.

(c) Classification accuracy.

(d) The size of source code, which is necessary for description of testing algorithm measured as number of source code lines.

Note that all the computation experiments reported with this study were carried out using CPU. It is planned that results obtained on the same tests using GPU will be published elsewhere later.

For estimation of scalability of examined frameworks, the above (a)-(c) metrics were measured for different scaling factors of FCNN networks. The following two kinds of scaling were applied to networks architecture:

- Varying the network's "depth" (number of internal layers) with fixed number of neurons in each layer (see Fig.2).
- Varying the network's "width" (number of neurons) with fixed number of layers (see Fig.3).

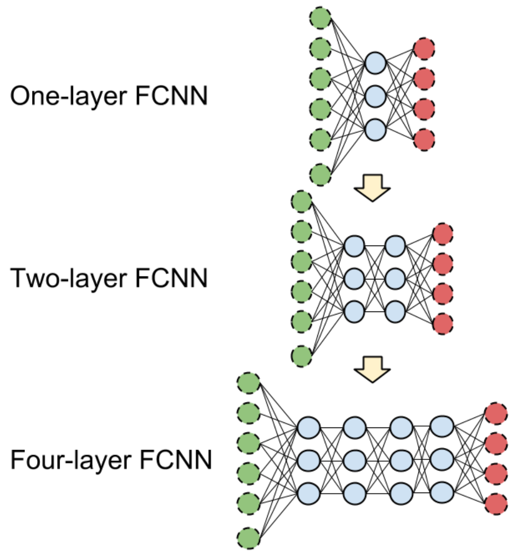

Fig.2 – Test-case #1: "in-depth" changes neural network architecture

In experiments with changing of network depth, the following values of control parameters were used:

- Number of internal layers: vary from 1 to 4.
- Number of neurons in internal layers: 100.

In experiments with changing of network width, the following parameters were set:

- Number of neurons in internal layers: subsequently set to 64, 128, 512 1024.
- Number of internal layers: 1.

Testing of neural networks was done using two different activation functions:

- Hyperbolic tangent (Tanh).
- Rectified linear unit function (ReLU) [10].

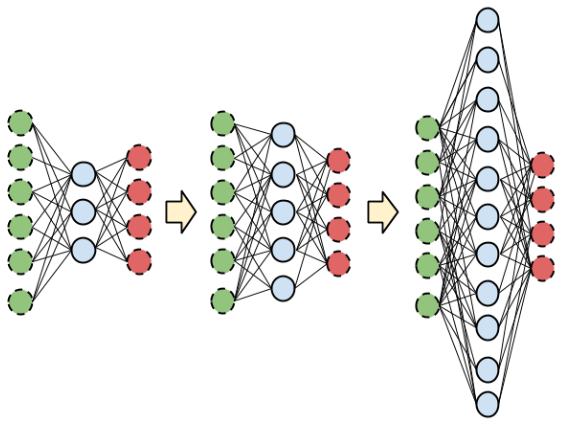

Fig.3 – Test-case #2: "in-width" changes of neural network architecture.

In order to compare results obtained with Tanh and ReLU functions, identical experiments have been performed and their results put side by side.

4. RESULTS

In Fig.4-Fig.6 below we present results of comparison of all five frameworks by the FCNN neural networks training time, the prediction time, and the resultant classification (i.e., digit image recognition) accuracy in case the Tanh nonlinearity function is employed. The number of epochs was set to 10 in all the experiments.

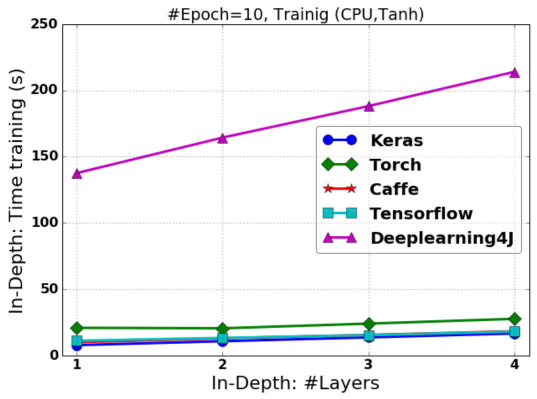

Fig.4 – Training time for FCNN architecture modification "in-depth" and Tanh nonlinearity function (lower is better).

As it can be seen from Fig. 4, the training time is kept within 30 seconds for all number of internal network layers ranging from 1 to 4 in all networks except for Deeplearning4J. In this particular framework the training time has started approximately from 140 seconds in case of 1 layer and permanently grows up to 210 seconds for 4 layers.

Similar pattern of computation expenses are observed for the prediction time as well (see Fig. 5) where Theano (with Keras wrapper), Torch, Caffe, and TensorFlow took less than 0.4 second of time on ordinary personal computer equipped with Intel i7 processor. In the same time, the Deeplearning4J started from 0.75 second in case of 1 layer and steadily increased up to almost 1.1 second for 4 layers.

Contrary to the pretty coherent behavior observed for training time, which is illustrated in Fig. 4-5, the classification accuracy percentages (see Fig. 6) demonstrate substantially different behavior making choice of the best Deep Learning framework more complicated. Indeed, this time Theano (Keras), Deeplearning4J, and Caffe have achieved reasonably high accuracy starting from 94% (Caffe, 1 layer) and going up to 98% (Theano-Keras, 4 layers). However, former leaders in time competition Torch and TensorFlow unpredictably drop their class prediction accuracy with increasing of the number of networks layers dramatically.

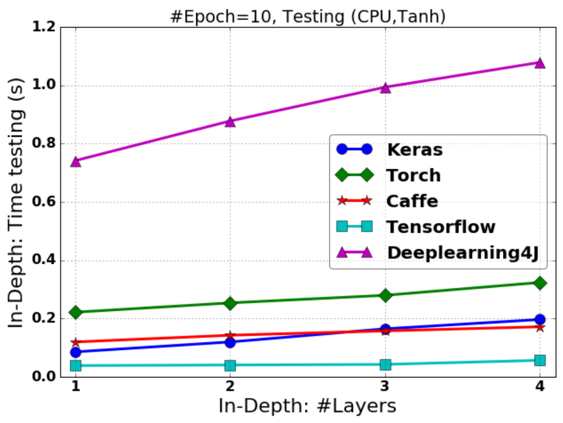

Fig.5 – Prediction time for FCNN architecture modification “in-depth” and Tanh nonlinearity function (lower is better)

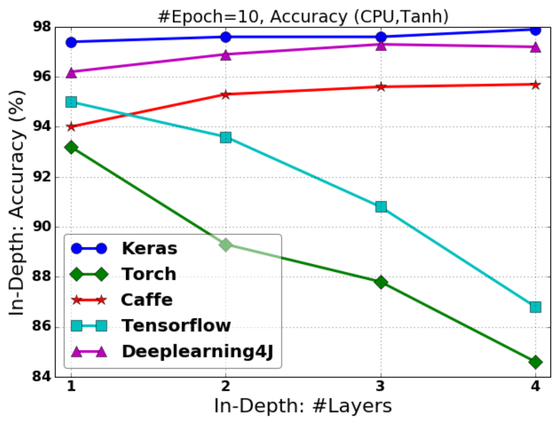

Fig.6 – Classification accuracy (in percents) for FCNN architecture modification “in-depth” and Tanh nonlinearity function (higher is better).

Similar to the testing results reported above, Fig.7, Fig.8, and Fig.9 below summarize results of comparison of all frameworks by their training time, the prediction time, and the resultant classification accuracy achieved with ReLU nonlinearity function.

Comparing plots of training time presented in Fig. 4 and Fig. 7 we can immediately infer that patterns of changes of training time depicted by curve shapes are very similar.

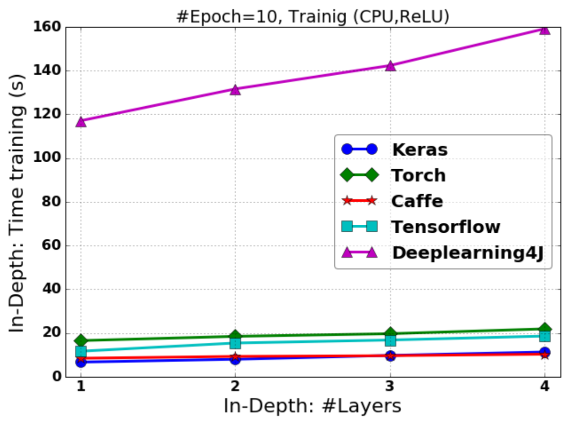

Fig.7 – Training time for FCNN architecture modification “in-depth” and ReLU nonlinearity function (lower is better)

However, the absolute values of training time in case of ReLU function is nearly twice lower (i.e., better) comparing the case of Hyperbolic tangent (Tanh) nonlinearity function. In contrast, despite some fluctuation in prediction time observed in Fig. 8, which illustrates the use of ReLU, in general, they are reasonably similar to those of Tanh (see Fig. 5).

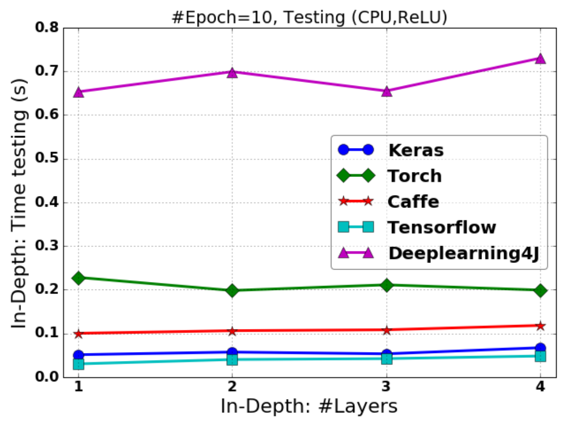

Fig.8 – Prediction time for FCNN architecture modification “in-depth” and ReLU nonlinearity function (lower is better)

Next, comparison of the final recognition accuracy provided by networks with Tanh (Fig. 6) and ReLU (Fig. 9) functions reveals that Torch framework changed its behavior cardinally (and more expectedly): its classification accuracy growing up now whereas other framework did not change their predictive abilities seriously.

Finally, since the use of ReLU function has demonstrated better results, the rest part of experiments on the assessing the influence of size of internal network layers was carried out with ReLU function only.

The final experiments were aimed at quantitative assessment of changes of speed and accuracy values associated with the size (i.e., number of neurons) in the hidden layer of networks like the ones depicted in Fig. 3 at the beginning of this section. Results of this last series of experiments presented in Fig. 10, 11, and 12 in the same way and order as it was before.

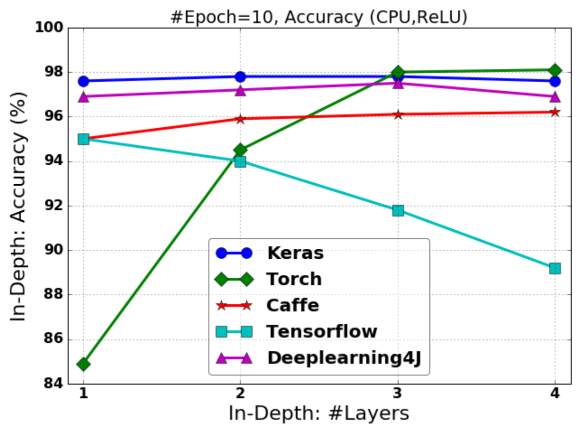

Fig.9 – Classification accuracy (in percents) for FCNN architecture modification “in-depth” and ReLU nonlinearity function (higher is better).

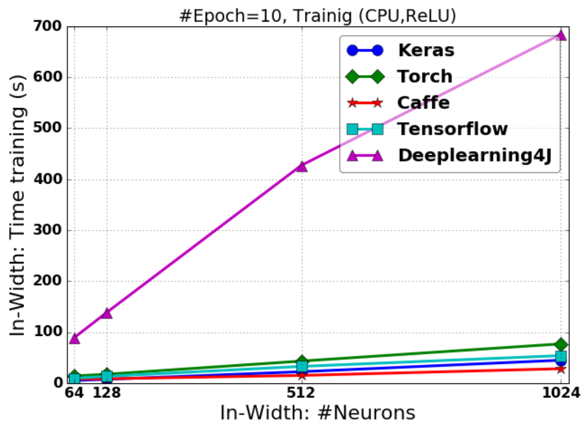

Fig.10 – Training time for FCNN architecture modification “in-width” and ReLU nonlinearity function (lower is better)

As it can be seen from Fig. 10, Deeplearning4J framework has demonstrated much longer time as compared to Theano (Keras), Torch, Caffe, and TensorFlow. In addition, training time of Deeplearning4J growing substantially with increasing number of neurons. In the same time, its competitors keep training time notably lower and almost constant.

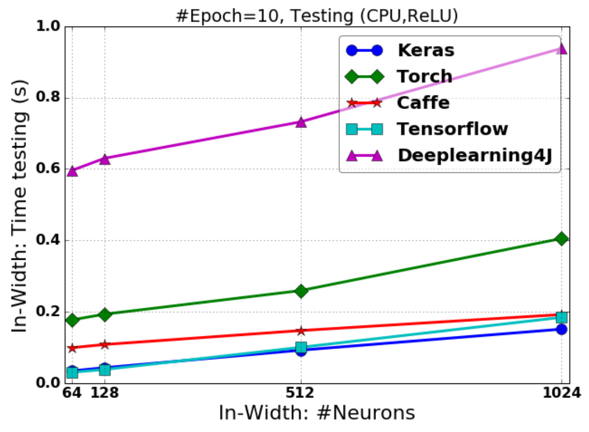

Fig.11 – Prediction time for FCNN architecture modification “in-width” and ReLU nonlinearity function (lower is better)

As for the prediction time (see Fig. 11), all the frameworks showed almost linear dependence on hidden layer size, which are, never the less, differ notably.

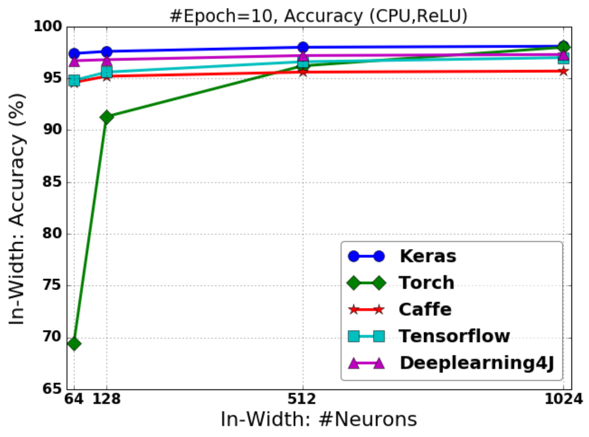

Fig.12 – Classification accuracy (in percents) for FCNN architecture modification “in-depth” and ReLU nonlinearity function (higher is better).

Surprisingly, the final classification accuracy of the major group of frameworks including Theano, Caffe, TensorFlow, and Deeplearning4J was kept high (around 97%) for all examined internal layer sizes ranging from 64 to 1024 neurons (Fig. 12). However, in case of Torch the classification accuracy expressed clear growth with the growing layer size.

The complexity of launching frameworks was measured in lines of source code needed to implement corresponding algorithm as well as their interface programming languages provided in Tab. 1 and Fig 13.

Table.1 – The framework complexity

DL Framework	Number of lines of code	Programming language
Theano (with Keras)	62	Python
Torch	165	Lua
Caffe	110	Protobuf
TensorFlow	80	Python
Deeplearning4J	130	Java

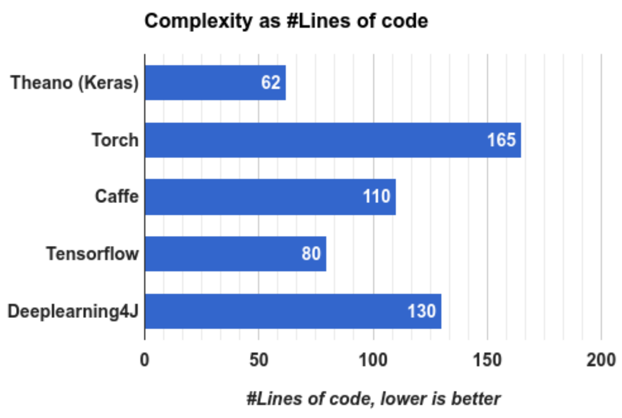

Fig.13 – Framework complexity, which measured as the number of lines of code needed to implement the algorithm (lower is better)

5. CONCLUSION

Results reported with this study allow drawing conclusions formulated below in a qualitative style. In case of any specific interest, corresponding quantitative values can be found in tables and plots above.

(1) As it can be predicted, deepening of the FCNN network, i.e., increasing of the number of network layers for improving the quality of classification leads to certain growth of computation expenses that is increases network training time.

(2) An increase in the number of neurons in hidden layer also requires more training time to achieve given level of classification accuracy.

(3) The use of non-linear activation function ReLU instead of Tanh increases network training speed and the resultant classification accuracy.

(4) During the testing of dependence of classification accuracy on the neural network depth (number of layers), Torch and TensorFlow frameworks have demonstrated somewhat strange behavior. With increase of network depth, their classification accuracy drops down. Such results can be explained either by incorrect computation of gradients during the network training or by the need for more sophisticated initialization of weighting coefficients and some in-depth knowledge of Torch framework.

(5) According to the results of testing of dependence of classification accuracy on the number of neurons in the internal layer of network, the five examined frameworks joined into 3 following groups:

- Caffe and Deeplearning4J have unexpectedly dropped in the accuracy with increasing number of neurons.
- Contrary to the previous group, TensorFlow and Torch have increased the classification accuracy with increasing number of neurons.
- Theano (with Keras) has demonstrated stable classification accuracy even with reasonably small number of neurons.

(6) Framework Deeplearning4J was the slowest in the network training and prediction. However, we need take into consideration that this framework is currently under development and its training and prediction speed may change in the near future.

(7) Finally, considering all the testing results on speed, classification accuracy and complexity of launching (number of lines of source code), the ranking list by these features will be as follows: Theano (with Keras), TensorFlow, Caffe, Torch, Deeplearning4J.

Having said the above, we need to remember that in general, it is impossible to provide fully objective comparison due to inevitable differences between the frameworks. For instance, counting the necessary lines of source code, we should keep in mind that the interfaces of frameworks under comparison use different programming languages (Python, Lua, Java), whose source lines are not equivalent.

Acknowledgements. This work was mainly funded by Altoros (www.altoros.com), a global provider of big data, cloud computing, and Platform-as-a-Service solutions. You can find the full version of this study at www.altoros.com/research-papers.

6. REFERENCES

[1] D.E. Rumelhart, G.E. Hinton, R.J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, vol. 5(3), 1988, p. 1.

[2] G. Cybenko. Approximations by superpositions of sigmoidal functions, *Mathematics of Control, Signals, and Systems*, vol. 2 (4), 1989, pp. 303-314.

[3] J. Schmidhuber. Deep Learning in Neural Networks: An Overview, *Neural Networks*, vol. 61, 2015, pp. 85–117.

[4] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra I. Goodfellow, A. Bergeron, Y. Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[5] R. Collobert, K. Kavukcuoglu, C. Farabet. Torch7: A matlab-like environment for machine learning. *In: BigLearn, NIPS Workshop* (No. EPFL-CONF-192376), 2011.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *In: Proceedings of the ACM International Conference on Multimedia*, ACM, 2014, pp. 675-678.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, S. Ghemawat. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467*, 2016.

[8] http://deeplearning4j.org/ Last visited 16.04.2016.

[9] http://yann.lecun.com/exdb/mnist/ Last visited 16.04.2016.

[10] X.Glorot, A.Bordes, Y.Bengio. Deep sparse rectifier neural networks. *In: International Conference on Artificial Intelligence and Statistics*, 2011, pp.315-323.