Discrete & Continuous Models
& Applied Computational Science
2019, 27 (4) 305–315
http://journals.rudn.ru/miph

Computer Science and Computer Engineering

Research article

UDC 519.21;51-76
DOI: 10.22363/2658-4670-2019-27-4-305-315

Review and comparative analysis of machine learning libraries for machine learning

Migran N. Gevorkyan, Anastasia V. Demidova, Tatiana S. Demidova, Anton A. Sobolev

Department of Applied Probability and Informatics
Peoples' Friendship University of Russia
6, Miklukho-Maklaya St., Moscow 117198, Russian Federation

(received: December 15, 2019; accepted: December 30, 2019)

The article is an overview. We carry out the comparison of actual machine learning libraries that can be used the neural networks development. The first part of the article gives a brief description of TensorFlow, PyTorch, Theano, Keras, SciKit Learn libraries, SciPy library stack. An overview of the scope of these libraries and the main technical characteristics, such as performance, supported programming languages, the current state of development is given. In the second part of the article, a comparison of five libraries is carried out on the example of a multilayer perceptron, which is applied to the problem of handwritten digits recognizing. This problem is well known and well suited for testing different types of neural networks. The study time is compared depending on the number of epochs and the accuracy of the classifier. The results of the comparison are presented in the form of graphs of training time and accuracy depending on the number of epochs and in tabular form.

Key words and phrases: machine learning, neural networks, MNIST, TensorFlow, PyTorch

1. Introduction

Due to the vast development of machine learning and data science, it is not possible to review the diversity of available software solutions. In this section we will consider only the most popular libraries and frameworks for neural networks development and machine learning.

The most common language for building neural networks at the moment is the Python language [1]. There is a number of reasons why this language has occupied this domain.

— Python is easy-to-learn language actively used in the field of school and university education. Because of this, it has gained popularity not only in industrial programming, but also among professionals who use programming as a research tool.
— The standard cpython interpreter makes it easy to create bindings for C-function calls, allowing Python to be used as a convenient interface for low-level libraries.
— The community has created a wide range of tools for interactive Python code execution and data visualization (e.g. [2]–[4]). Especially it is useful for scientific research, where almost always there is no original clear algorithm of solutions and it is necessary to conduct a scientific search.

A significant disadvantage of Python is its low performance, which can be overcome by writing critical parts of software in a compiled language (it is usually C or C++) or by using cython [5] translator.

Many machine learning libraries are also written in two or more languages. The part of software, which handles main part of computations, is usually implemented in C or C++ (the *core* part or *backend*). Pure Python is used for bindings to organize a convenient and easy to use interface (*interface part* or *frontend*). So, if a library is implemented in pure Python, then in most cases:

— It is an add-on to another, lower-level library and provides a more user-friendly and easy-to-learn interface;
— It is designed for educational purposes or for prototyping.

Note also that all the libraries in this review are free open source software. Also note that Python 2 support will be discontinued from the beginning of 2020 for vast majority of python libs.

2. Overview of machine learning (ML) libraries

From all reviewed libraries, only TensorFlow and PyTorch directly compete with each other. Other libraries complement each other's functionality and specialize in their own area.

2.1. Scientific Python (SciPy)

Scientific Python libraries set is not directly related to machine learning, but many machine learning libraries rely on Scientific Python components in their work. Let us briefly describe the main components included in this set.

— NumPy [6] is the library which implements high performance arrays and tools for them. The computational core is written in C (52% of code base) with the interface part in Python (48% of code base). Linear algebra functions heavily rely on LAPACK library. NumPy implements variety of linear algebra methods for working with vectors, matrices and tensors (multidimensional arrays in this case). It also supports parallel computing by utilizing vector capabilities of modern CPUs.
— SciPy [7] is the library that implements many mathematical methods, such as algebraic equations and differential equations solvers, polynomial

interpolation, various optimization methods, etc. For a number of methods the Fortran (about 23% of the total code base) and C (20% of the code base) libraries are used.

— Pandas [8] is the library designed to work with time series and table data (DataFrame data structure). It is written almost entirely in pure Python using NumPy arrays and is often used in machine learning to organize training and test samples.

In addition to these three main libraries, the scientific Python stack also includes Matplotlib [3] for data visualization and plotting, and a set of interactive shells, such as iPython and Jupyter [2].

2.2. TensorFlow

TensorFlow [9], [10] is an open source library used primarily for deep machine learning. It was originally developed on by Google's divisions, but in 2015 it was released as free open source software under the Apache License 2.0. The current stable version is 2.1.0.

The computational core is written in C++ (60% of all code) using CUDA technology, which allows one to utilize graphics cards in calculations. The interface part is implemented in Python (30% of all code base). There are also unofficial bindings for other languages, but only C++ and Python interfaces are officially supported.

The library is based on the principle of *data flows* (dataflow), according to which the program is organized in the form of computational blocks associated with each other in the form of a directed graph which is called *computational graph*. Data is processed by passing from one block to another.

Such application architecture makes it easy to use parallel calculations on both multi-core CPUs and distributed cluster systems. In addition, it is well suited for building neural networks in which each neuron is presented by an independent component.

In addition to the computational graph, TensorFlow uses a data structure called *tensor*. It is similar to the tensor from differential geometry in the sense that it is a multidimensional array.

2.3. PyTorch

The PyTorch [11] library was created on the basis of Torch [12]. The original Torch library was developed in C and used Lua as the interface. With the growth of Python popularity in machine learning, Torch has been rewritten in C++11/CUDA (60% code) and Python (32% code). Initial development was conducted in the company of Facebook, but currently PyTorch is an OpenSource library, distributed under a BSD-like license. The current version is 1.3.1.

PyTorch, as well as TensorFlow, is built on the basis of dataflow concept. The main difference from TensorFlow is that in TensorFlow computational graph is static, then in PyTorch the graph is dynamic. This means that one can modify the graph on the fly, adding or removing nodes as needed. In TensorFlow, the entire graph must be specified before the model run.

The developers of PyTorch emphasize that Python is tightly integrated into the library (library is more pythonic). This makes it easier to use than

TensorFlow, as the user does not have to dive into low-level parts written in C++.

It is worth noting, however, that TensorFlow surpasses PyTorch in popularity, as it appeared earlier and is used in many educational courses on machine learning.

2.4. Theano

Theano [13] library is a Python interface for the optimizing compiler. It allows user to specify functions and after that translates them to C++. Then Theano compiles C++ code to run it on the CPU (using g++ for compilation), or on the graphics accelerator (using nvcc to utilize CUDA). In addition, automatic differentiation algorithms are built into the library.

After optimization and compilation, the functions become available as regular python functions, but have high performance. Vector, matrix, and tensor operations are supported and efficiently parallelized on available hardware (multi-core processor or graphics accelerator).

With support for multidimensional array operations and automatic differentiation, Theano is widely used as a backend for building neural networks. In particular, it can be used by the Keras library.

Theano is written almost entirely in Python, but requires NumPy, SciPy, pyCUDA and BLAS, as well as g++ or NVIDIA CUDA compilers (recommended for optimal performance).

Development of the library was suspended in 2017, but resumed in 2018. The current version is 1.0.4.

2.5. Keras

The Keras [14] library provides a high-level programming interface for building neural networks. It can work on top of TensorFlow, Microsoft Cognitive Toolkit (CNTK) [15] or Theano [13]. The library is written entirely in Python and is distributed under the MIT license. Current version 2.3.1

The library is based on the following principles: ease of use, modularity, extensibility.

The modularity principle allows you to separately describe the neural layers, optimizers, activator functions, etc, and then combine them into a single model. The model is fully described in Python. The created model can be saved to disk for further use and distribution.

2.6. SciKit Learn

SciKit Learn [16] is the library for data processing. It implements various methods of classification, regression analysis, clustering and other algorithms related to classical machine learning. It is written almost entirely in Python (98% of all code base), but uses NumPy and SciPy for algorithms implementation. Despite the fact that number of current version is 0.21.1, the project is very stable, as it has been developing since 2007.

SciKit Learn is suitable for traditional machine learning and data preprocessing tasks. This library does not support the concept of dataflow and

does not allow one to create his own models. The absence of a computational graph does not allow flexible scale of models for multi-core processors and graphics accelerators and forces to limit the degree of parallelism that is implemented in NumPy.

3. Comparative analysis of machine learning libraries

3.1. Description and architecture of neural network

For the comparative analysis of deep machine learning libraries, we choose the problem of handwritten digit recognition from the MNIST database and the neural network [17] to solve it.

The MNIST database is contained in a CSV file, where comma-separated digits are written. In a CSV file, the first value is a marker that represents the corresponding digit. Next value is the size of the digit image in pixels, consisting of 784 values and having a dimension of square 28x28.

The training file consists of 60 thousand copies, and the test file of 10 thousand copies. To solve the problem we choose the MLP (multilayer perceptron) architecture. Perceptron was one of the first models of neural networks, which was supposed to simulate the neural processes in human mind. This model was proposed by Frank Rosenblatt in 1957 and first implemented in 1960 [18]. A multilayer perceptron according to Rosenblatt differs from a single layer in that it contains additional hidden layers.

The neural network (Figure 1) consists of an input layer, two hidden layers, and one output layer. The input layer contains 784 neurons, the hidden layers contain 256 neurons, and the output layer 10, according to the number of features. The activation function in hidden layers is ReLU, which has the form $f(x) = \max(0, x)$ [19]. Stochastic gradient descent (SGD) [20] is used as the optimization algorithm.

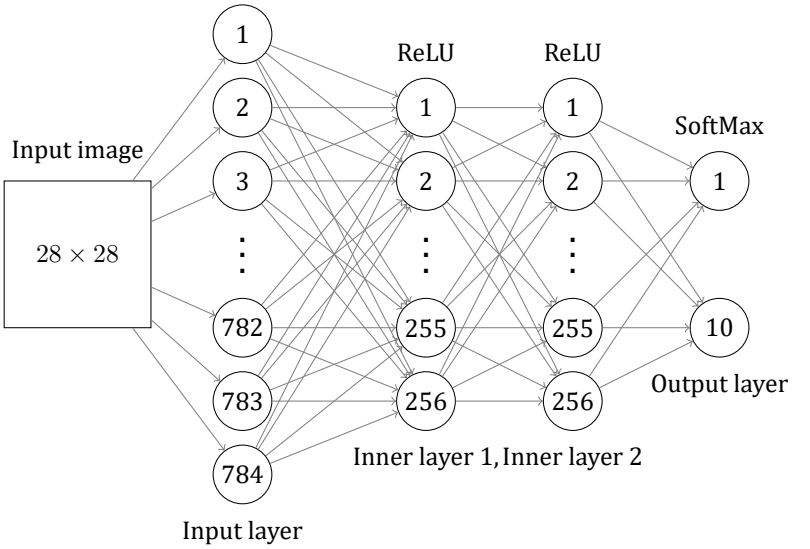
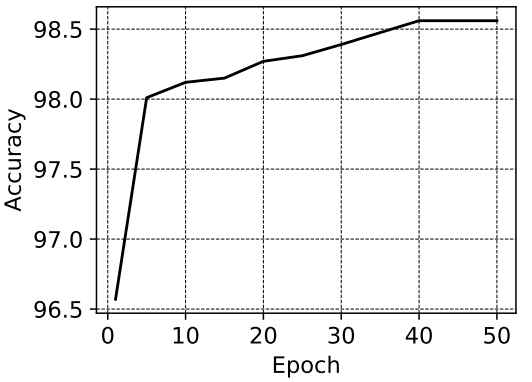


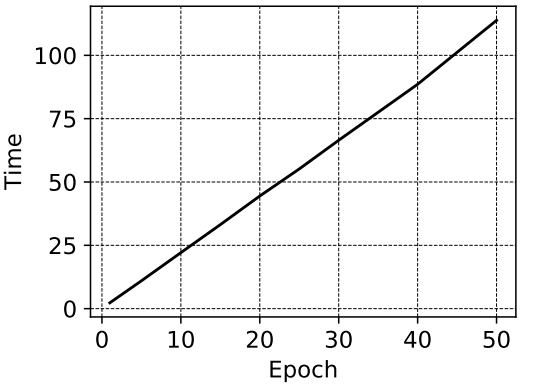
Figure 1. A multilayer perceptron for the recognition of handwritten digits

3.2. Software implementation of a neural network using various libraries

With each library from the overview above, we built perceptron models. Each neural network was trained, and training time and accuracy were measured for a different number of training eras. We used these measurements for comparative analysis of libraries. The constructed graphs describe the dependence of learning time and accuracy on the number of eras (Figures 2–6).

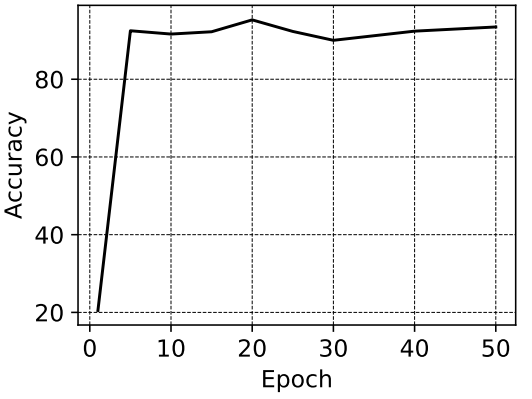


(a) The dependence of accuracy on the number of epochs

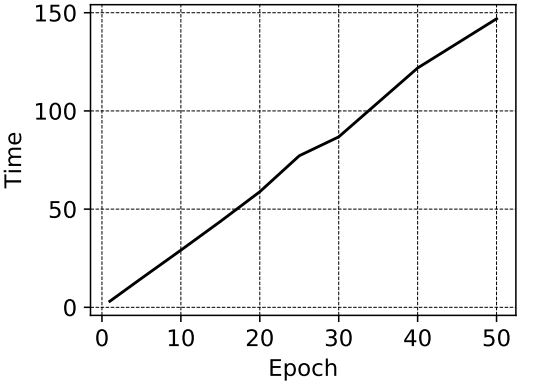


(b) The dependence of training time on the number of epochs

Figure 2. Results of a neural network built with the help of the Keras library



(a) The dependence of accuracy on the number of epochs



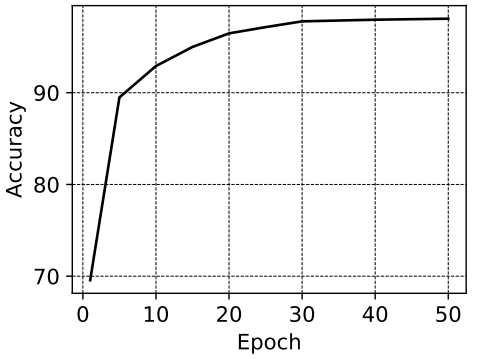
(b) The dependence of training time on the number of epochs

Figure 3. Results of a neural network built with the help of the SciKit Learn library

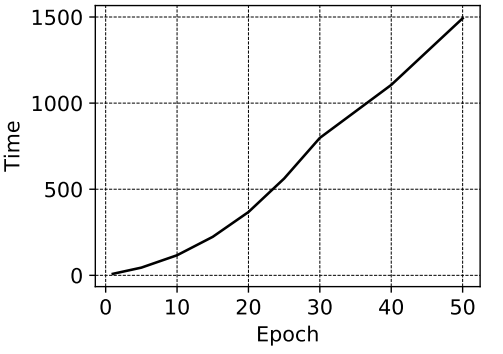
The first plot shows the dependence of time on the number of epochs. The second plot shows the dependence of accuracy on the number of epochs.

Below is the summary table of the results of neural network training at 50 epochs for different libraries (see Table 1).

On the diagram (Figure 7) the time and accuracy values are shown for the considered libraries. All values are normalized.

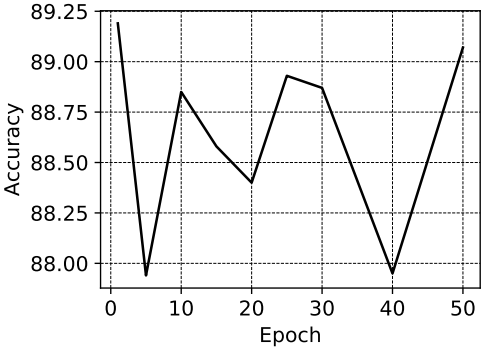


(a) The dependence of accuracy on the number of epochs

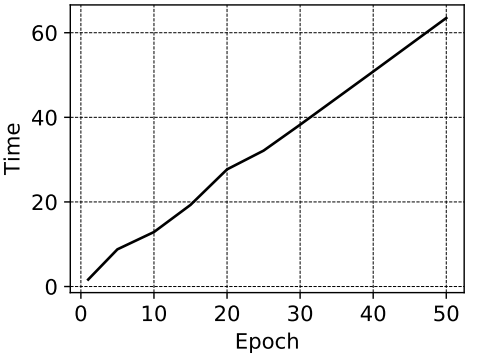


(b) The dependence of training time on the number of epochs

Figure 4. Results of a neural network built with the help of the PyTorch library

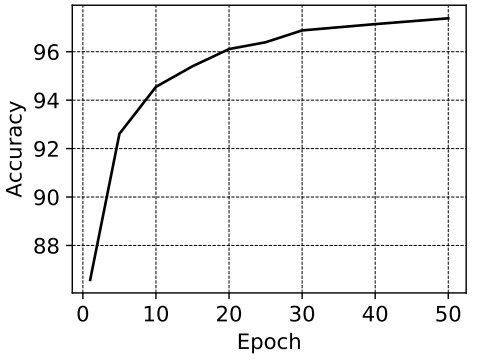


(a) The dependence of accuracy on the number of epochs

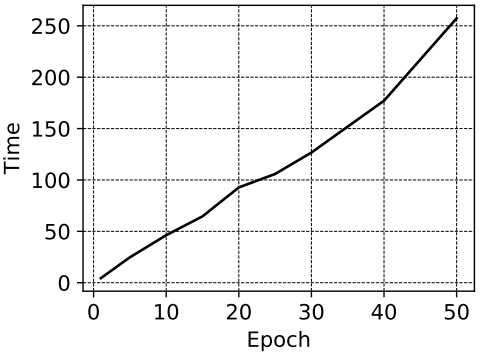


(b) The dependence of training time on the number of epochs

Figure 5. Results of a neural network built with the help of the TensorFlow library



(a) The dependence of accuracy on the number of epochs



(b) The dependence of training time on the number of epochs

Figure 6. Results of a neural network built with the help of the Theano library

Table 1

Results of comparative analysis of machine learning libraries

Library	Accuracy, %	Time, sec.
MLPClassifier	93.45	146.90
Keras	98.56	113.80
TensorFlow	89.07	63.48
Theano	97.38	257.29
PyTourch	98.07	1492.29

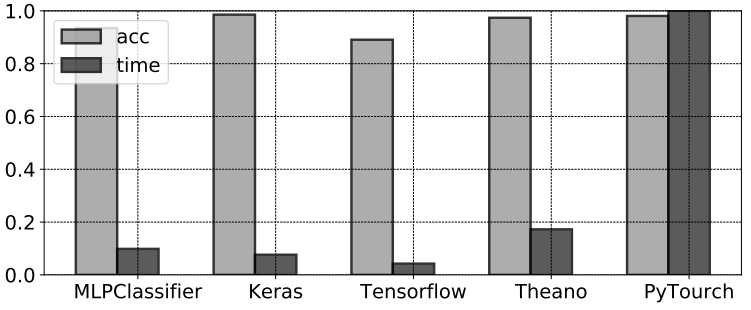


Figure 7. The time and accuracy values

In the PyTorch library, the learning time increases faster than in all other libraries with the growth of epochs, which time is approximately the same. The learning time of TensorFlow, Scikit-learn and Keras libraries varies from 1 to 3 seconds per epoch. While this indicator in PyTorch exceeds 8 seconds, which is several times higher than the training time of other libraries.

The accuracy of the TensorFlow library does not exceed 0.9, which is low compared to other libraries. Scikit-learn also showed a low accuracy result. The PyTorch library shows a good accuracy result only with a large number of epochs, but with the growth of the number of epochs, the learning time increases greatly. The minimum accuracy was 98.07. The Keras and Theano libraries are the most accurate and their accuracy is kept at 0.98.

4. Conclusion

Based on the comparison of different libraries, a number of conclusions can be drawn. Almost all libraries except PyTorch show approximately the same learning time. In the case of PyTorch, the longer learning time can be explained by the support of a dynamic computational graph, which apparently imposes additional computational costs. In turn, the TensorFlow library showed an average accuracy result, behind PyTorch and Theano.

Acknowledgments

The publication was prepared with the support of the “RUDN University Program 5-100”.

References

[1] G. Van Rossum and F. L. Drake Jr, *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[2] T. Kluyver *et al.*, "Jupyter Notebooks — a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds., IOS Press, 2016, pp. 87–90. DOI: 10.3233/978-1-61499-649-1-87.

[3] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[4] F. Pérez and B. E. Granger, "IPython: a system for interactive scientific computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21–29, May 2007. DOI: 10.1109/MCSE.2007.53.

[5] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: the best of both worlds," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 31–39, Mar. 2011. DOI: 10.1109/MCSE.2010.118.

[6] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011. DOI: 10.1109/MCSE.2011.37.

[7] E. Jones, T. Oliphant, P. Peterson, *et al.* (2001–). SciPy: open source scientific tools for Python, [Online]. Available: http://www.scipy.org/.

[8] W. McKinney, "Data structures for statistical computing in Python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51–56.

[9] Martín Abadi *et al.* (2015). TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org, [Online]. Available: http://tensorflow.org/.

[10] (2019). TensorFlow official repository, [Online]. Available: https://github.com/tensorflow/tensorflow.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*, 2017.

[12] (2019). Torch official repository, [Online]. Available: https://github.com/torch/torch7.

[13] Theano Development Team, "Theano: a Python framework for fast computation of mathematical expressions," May 2016. eprint: arXiv:abs/1605.02688.

[14] F. Chollet. (2019). Keras, [Online]. Available: https://keras.io/.

[15] (2019). CNTC official repository, [Online]. Available: https://github.com/Microsoft/cntk.

[16] F. Pedregosa *et al.*, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] (2019). MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges, [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[18] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, 1958. DOI: 10.1037/h0042519.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 315–323.

[20] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04, Banff, Alberta, Canada: ACM, 2004. DOI: 10.1145/1015330.1015332.

For citation:

M. N. Gevorkyan, A. V. Demidova, T. S. Demidova, A. A. Sobolev, Review and comparative analysis of machine learning libraries for machine learning, Discrete and Continuous Models and Applied Computational Science 27 (4) (2019) 305–315. DOI: 10.22363/2658-4670-2019-27-4-305-315.

Information about the authors:

Migran N. Gevorkyan — Candidate of Physical and Mathematical Sciences, assistant professor of Department of Applied Probability and Informatics of Peoples' Friendship University of Russia (RUDN University) (e-mail: gevorkyan-mn@rudn.ru, phone: +7(495)9520250, ORCID: https://orcid.org/0000-0002-4834-4895, ResearcherID: E-9214-2016, Scopus Author ID: 57190004380)

Anastasia V. Demidova — Candidate of Physical and Mathematical Sciences, assistant professor of Department of Applied Probability and Informatics of Peoples' Friendship University of Russia (RUDN University) (e-mail: demidova-av@rudn.ru, phone: +7(495)9520250, ORCID: https://orcid.org/0000-0003-1000-9650, ResearcherID: AAD-2214-2019, Scopus Author ID: 57191952809)

Tatiana S. Demidova — student of Department of Applied Probability and Informatics of Peoples' Friendship University of Russia (RUDN University) (e-mail: 1032152607@pfur.ru, phone: +7(495)9520250, ORCID: https://orcid.org/0000-0001-6076-1619, Scopus Author ID: 57210151914)

Anton A. Sobolev — student of Department of Applied Probability and Informatics of Peoples' Friendship University of Russia (RUDN University) (e-mail: 1032152618@pfur.ru, phone: +7(495)9520250, ORCID: https://orcid.org/0000-0003-1629-1378, Scopus Author ID: 57206139419)

УДК 519.21;51-76
DOI: 10.22363/2658-4670-2019-27-4-305-315

Обзор и сравнительный анализ библиотек машинного обучения для построения нейронных сетей

М. Н. Геворкян, А. В. Демидова, Т. С. Демидова, А. А. Соболев

Кафедра прикладной информатики и теории вероятностей
Российский университет дружбы народов
ул. Миклухо-Маклая, д. 6, Москва, 117198, Россия

Статья носит обзорный характер. В ней проведено сравнение актуальных библиотек машинного обучения, которые могут быть использованы для построения нейронных сетей.

В первой части статьи даётся краткое описание библиотек TensorFlow, PyTorch, Theano, Keras, SciKit Learn, стека библиотек SciPy (NumPy, SciPy, Pandas, Matplotlib, Jupyter). Делается обзор области применения перечисленных библиотек и основных технических характеристик, таких как быстродействие, поддерживаемые языки программирования, текущее состояние разработки. Среди рассматриваемых библиотек только PyTorch и TensorFlow непосредственно конкурируют друг с другом. Остальные библиотеки взаимодополняют друг друга и часто используются совместно при построении различных моделей машинного обучения.

Во второй части статьи проводится сравнение пяти библиотек на примере многослойного перцептрона, который применяется к задаче распознания рукописных цифр. Данная задача хорошо разработана и является модельной для тестирования различных реализаций нейронных сетей. Сравнивается время обучения в зависимости от количества эпох и точности работы классификатора. Результаты сравнения представлены в виде графиков времени обучения и точности в зависимости от количества эпох и в табличном виде.

Ключевые слова: машинное обучение, нейронные сети, MNIST, TensorFlow, PyTorch