Thieme

# Developing Deep Learning Applications for Life Science and Pharma Industry

Authors
**Daniel Siegismund[1], Vasily Tolkachev[2], Stephan Heyse[1], Beate Sick[2], Oliver Duerr[2], Stephan Steigele[1]**

Affiliations
1 Genedata AG, Basel, Switzerland
2 Institute of Data Analysis and Process Design, Winterthur, Switzerland

Key words
drug research, imaging, screening

**received** 20.06.2017
**accepted** 08.12.2017

Bibliography
**DOI** https://doi.org/10.1055/s-0043-124761
Published online: 16.1.2018
Drug Res 2018; 68: 305–310

ISSN 2194-9379

Correspondence
Stephan Steigele
Genedata AG
Margarethenstrasse 38
CH-4053 Basel
Switzerland
Tel.: +41/61/511 8423, Fax: +41/61/511 8484
stephan.steigele@genedata.com

**ABSTRACT**

Deep Learning has boosted artificial intelligence over the past 5 years and is seen now as one of the major technological innovation areas, predicted to replace lots of repetitive, but complex tasks of human labor within the next decade. It is also expected to be 'game changing' for research activities in pharma and life sciences, where large sets of similar yet complex data samples are systematically analyzed. Deep learning is currently conquering formerly expert domains especially in areas requiring perception, previously not amenable to standard machine learning. A typical example is the automated analysis of images which are typically produced en-masse in many domains, e. g., in high-content screening or digital pathology. Deep learning enables to create competitive applications in so-far defined core domains of 'human intelligence'. Applications of artificial intelligence have been enabled in recent years by (i) the massive availability of data samples, collected in pharma driven drug programs ( = 'big data') as well as (ii) deep learning algorithmic advancements and (iii) increase in compute power. Such applications are based on software frameworks with specific strengths and weaknesses. Here, we introduce typical applications and underlying frameworks for deep learning with a set of practical criteria for developing production ready solutions in life science and pharma research. Based on our own experience in successfully developing deep learning applications we provide suggestions and a baseline for selecting the most suited frameworks for a future-proof and cost-effective development.

## Introduction

Deep learning methods are based on artificial neural networks (NN) with multiple hidden layers between the input signal and the result (aka "deep networks"). The introduction of 'deep' multilayered artificial neural networks has drastically improved performance for tasks like automatic speech recognition, natural language recognition or computer vision [1]. While the theoretical concepts for deep learning were established in the 1980s to 1990s [2]–[3], their applicability to real world problems was dependent on potent computer hardware, like graphic cards, or high performance computer clusters that only became available in the 2000s. A strong increase in digitization, and thus massive availability of data and images, also contributed to their success and recent advances have elevated deep learning to surpass human performance in tasks like playing board games [4] or in the classification of images [5].

The most frequently used network architectures for image recognition are the so-called convolutional neural networks (CNNs). They try to mimic the way the human brain classifies objects into classes based on learned features that are combined in class-specific representations. The CNN learns during training which features need to be extracted from an image to solve the given classification problem (▶ **Fig. 1**). In imaging, the first participation of a CNN in the large scale image classification challenge ImageNet in 2012 was an important milestone, since it dwarfed the performance of all conventional machine learning approaches [6]. After this first breakthrough CNNs have proven their applicability in different use

cases all over the imaging domain [1]. Also in other commercially important domains such as speech recognition a leap in performance has been achieved by deep learning methods [7].

The evolution of stacked neural networks for machine learning tasks (deep learning) started in academic research, resulting in Theano as first generally available deep learning software framework [8]. However, the recent development in this field is driven by companies like Google, Facebook, Microsoft and NVIDIA. These companies also provide the hardware and respective software packages. A downside of having many contributors is a multitude of different frameworks which can be used to set up a deep learning implementation, each with its own advantages and disadvantages. In addition, frameworks that are suitable for academic research purposes may not well reflect the needs of the pharma industry research process.

### Applications in Life-Sciences and Pharmaceutical Research

Deep learning methods showed utility and strong benefits when applied in a large panel of different life science and pharma applications. Three main areas benefit so far the most: Chemo-Informatics, Computational Genomics and Biomedical Imaging [9]–[10].

### Chemo-Informatics

Chemo-Informatics is a vital field of modern pharma research enabling a unique view for understanding of molecular interactions, for finding of lead structures and to enlarge lead series in virtual screening. The goal of virtual screening is the prediction of the binding potential of small molecules to (drug) targets. Thus, virtual screening acts as pre-filter for the selection of molecules to be tested in real world experiments. Especially structure based virtual screening approaches based on deep learning had been successfully applied recently, e. g., to identify ligands of target proteins [11] or for docking-based experiments with over different 40 proteins achieving the best results in a tested panel of 26 algorithms [12]. On a much simpler scale, deep learning approaches also have shown superior performance for secondary structure prediction in peptides [13].

Quantitative structure–reactivity relationship prediction relates the molecular structure to its chemical properties and biological effects. Deep learning has gained a huge influence in this area, showing better performance than conventional machine learning methods while much less prior knowledge is required for training of predictive models [14]. A very successful application of deep learning with already unmatched performance is quantitative structure–property relationship (QSPR) predictions, in particular for toxicology where recently a deep learning approach won the computational Tox21 data challenge [15].

### Computational Genomics

Application of deep learning in computational genomics has been reported for gene expression analysis and for structural genomics. One influential application in the field of gene expression analysis had been the use of transcriptomic data to predict the therapeutically relevant regulatory pathways of active compounds from data obtained on many cell lines, outperforming conventional machine learning approaches [16]. Genome-wide gene expression analysis is nowadays cheap per genome, but still costly when performed in parallel on thousands of samples. In silico gene expression inference by deep learning showed highly promising results here [17] and may be on option in closing experimental gaps in large scale studies by inferring missing expression data.

A deep understanding of regulation in biological systems and for identifying causal connections between disease states and their development requires knowledge about epigenetic or direct transcriptional regulation. In both areas deep learning showed already outstanding performance compared to existing computational methods, e. g., for accurate prediction of DNA methylation (CpG coverage) on a single cell methylation [18] or to predict transcription factor binding site motifs of DNA or RNA binding proteins [19].

### Biomedical Imaging

In biomedical imaging, deep learning approaches have been applied successfully to computer vision like cell segmentation and classification in High-Content Screening [20]. Deep learning in general and CNNs in particular are well suited for applications that produce a lot of (image) data, since they require a high number of training data. Thus, in particular for the imaging domain of pharma research the following areas seem to be a perfect fit for deep learning:

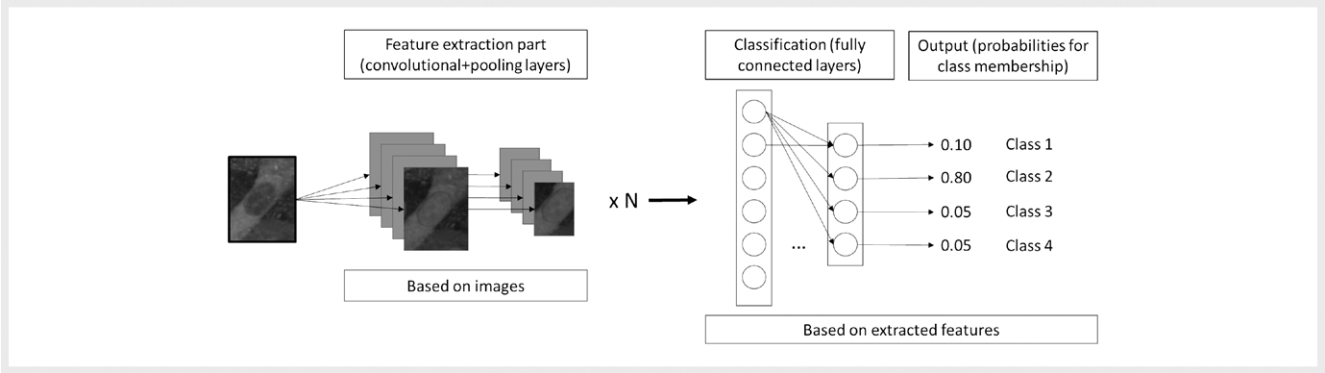
a. Automated microscopy of cells in high-content screening e. g., during early drug discovery, phenotypic classification in high-content screening experiments [20–21]
b. Digital histopathology of tissues produced by automated slide scanners e. g., in preclinical safety studies (tox-pathology); cell segmentation of histopathology and microscopy images [22], classification of tissue in histopathology [23].
c. Behavioral studies of mammals with video recording e. g., in preclinical development [24].
d. Light sheet and lattice light sheet fluorescence microscopy with high temporal and spatial resolution e. g., for toxicity studies in preclinical development

CNN approaches have proven the versatility to deal with different types of images/videos and thereby answering different research questions. In all circumstances CNNs are outperforming conventional methods by far and seem to be more robust with regard to e. g., image noise and other types of distortions.

## Deep Learning Frameworks: General Considerations

Important breakthroughs in deep learning occurred just in the recent past due to algorithmic and hardware developments producing real world applications which deliver drastically improved performance in object recognition, detection and classification tasks [6, 25]. Access to this technology is therefore relatively easy and is supported by development frameworks which enable software developers to create CNN based solutions without having to understand & implement deep learning hardware specific settings.

All of the frameworks that we consider work on NVIDIA graphic cards and use their programming interfaces: CUDA; the interface for AMD graphic cards, OpenCL is supported in a few frameworks only. A number of frameworks allow to use multiple GPUs on a sin-




▶ **Fig. 1** Schematic representation of a deep neural network architecture typical for image classification. An image (left) is processed by the network through a sequence of N sets of one convolutional and pooling layer each. Finally, two fully connected layers deduce classification information on increasing abstraction levels and yield class probabilities for objects in the image (right). The left network section aims to extract features, while the right section performs the classification based on feature vectors per image object. The rightmost layer is the output layer consisting of as many nodes as there are classes learned during the training. Per input image, a probability of membership in each class is calculated. Typically the image is classified to the label of highest probability.

gle server in parallel, or even distribution of computations between different GPU cluster nodes. Support across different hardware platforms and for specialized file formats like HDF5 [26] can become important in terms of possible scaling effects and for handling of huge data sets which do not fit in a computer RAM.

Deep learning is a fast-moving field and new inventions are being continuously made. All frameworks support the recent convolutional architectures and exhibit quite a similar performance, since they utilize the CUDA / openCL interface for computation. The situation is a bit different when it comes to special network architectures like recurrent neural networks (RNNs) [27]. These special architectures are important for many applications on time-resolved data, an important case in the pharma research context. Finally, the user needs to learn using the respective frameworks: therefore, well-documented tutorials and an increasingly active user base provide additional help and are a valuable asset for the beginners in the field.

## Selecting a Deep Learning Framework for an Automated Analysis Application Pharma / Life Sciences Application

Wikipedia lists more than 15 independent framework projects, most of them open source software [28]. Which of those are best suited for imaging applications in pharma settings? In our assessment we focus on the six biggest framework projects which show a sufficiently large user base, a license model without additional cost and industry-ready deployment possibilities.

## Criteria for Deep Learning Framework Assessment

There are several software quality models to define software quality from a theoretical point of view including models of the international organization for standardization (ISO) [29], or models tailored for open-source software e. g., the SQO-OSS model [30]. While those criteria are used to evaluate software projects (e. g., for the Linux Kernel), we found them less applicable for selecting a framework to develop deep learning applications. Therefore, we developed our own schema for selecting the best framework covering the requirements listed above:

1. Active Framework Development (closed vs. open source, speed of bug fixing and addition of new features), Software License (does it allow commercial usage?),
2. Age/Maturity (assuming that projects of decent age have major bugs resolved already),
3. Support (commercial support available?, strong user community available and responsive?) and
4. Deployment (does it work on different hardware architectures, does it scale when moving from single client to server to cloud based applications).

## Overview of Deep Learning Frameworks

▶ **Table 1** shows our selection of the six deep learning frameworks that may be explicitly used for production applications and are evaluated by the criteria that were discussed above.

### Theano

Theano is the pioneering deep learning framework, and was established early, in 2008 by the MILA lab at the Université de Montréal [8]. It has a very mature code base with a well-established interface. It continues to be well adopted by the (academic) community and the development is active. The performance is on par with other frameworks [31], and its underlying structure utilizes a very elegant way of apply complex calculations on datasets (graph approach), which allow similar flexibility to TensorFlow.

### Caffe

Caffe is also a mature deep learning framework, established 2013 by the Berkeley Vision and Learning Center at UC Berkeley. It is widely accepted in the academic community, especially in the im-

▲ **Table 1** Overview of deep learning frameworks.

| Framework | Creator | Software license | Programming language (interface) | Wrappers | Activity of development (GitHub commits last half year / # forks) | Age / Initial release | Support | Deployment * |
|---|---|---|---|---|---|---|---|---|
| TensorFlow | Google | Apache 2.0 | Python | TFlearn, scikit flow, Keras | 5614 / 35483 | Nov 2015 | Community | Yes with distributed TensorFlow |
| Caffe | BVLC, University of California, Berkeley | BSD | Python | - | 16 / 12583 | Sep 2013 | Community | Not fully |
| Theano | Université de Montréal | BSD | Python | Lasagne, Keras | 1534 / 2309 | Jan 2008 | Community | Yes, experimentally |
| DeepLearning4j | Skymind | Apache 2.0 | Java | - | 1108/ 3667 | Aug 2015 | Community and Commercial | Yes with native Apache SPARK support |
| Neon | Nervana Systems (now Intel) | Apache 2.0 | Python | - | 68 / 717 | Oct 2015 | Community and Commercial | Yes only in Nervana Cloud |
| MXNET | Various Universities | Apache 2.0 | Various (e. g., Python, R) | - | 905 / 4261 | 2015 | Community | Yes |

( * Support of different parallel architectures - multi GPU or cluster)

The six most relevant networks are rated according the criteria as explained in the text above. Wrappers provide a high level interface to the frameworks. Activity of development is measured by GitHub commits and forks: A GitHub commit is a change in the software that the active developers make and indicates the activity of development. A GitHub fork is a copy of the framework repository and is an indication of the popularity of the software through the community.

aging domain. Of advantage are the big 'model zoo' with many pre-trained models and multi-platform compliance. However, it lacks state-of the art network architectures like Recurrent Neural Networks RNNs [32]. In addition, the computing speed is not on par with the more modern frameworks [33].

## Neon

Neon is a new framework released at the end of 2015 by Nervana. It is one of the fastest frameworks, performing exceptionally well in imaging applications using CNNs [31, 33]. However, it is not active developed and still lack features that are part of other frameworks. The recent development of dedicated deep learning hardware called the Nervana Engine, sounds promising, but with the acquisition of Nervana by Intel the future development is unclear.

## MXnet

MXnet was created in a joint effort of researchers from different universities and is also one of the most performant frameworks. It works very resource-efficiently with a limited GPU and RAM memory, and scales almost linearly with the number of GPUs [34]. It operates with a large number of languages, such as Python, R, Matlab, Lua, C++, Scala, JS, Julia, and GO, which all use C++ as a backend for executing the computational graph. MXnet provides the possibility to mix different programming paradigms and uses a similar graph structure for calculations as Theano or TensorFlow. Optimization and parallelization in multi-GPU and cluster setups is automatic. It provides cutting-edge features such as the possibility to use Long Short-Term Memory (LSTM) or other recurrent neural network (RNN) architectures as well as the newest CNN architectures such as ResNet.

## TensorFlow

TensorFlow is a deep learning framework developed by the Google Brain Team. Since its initial release in November 2015, the framework had a huge impact on the deep learning research community. It is backed by professional development at Google and is thus the most actively developed framework with around 5614 GitHub commits/half year. Cutting-edge features and functionalities are adopted early, e. g., the possibility to use recurrent networks. In addition TensorFlow uses a graph structure similar to Theano and is able to deal with parallelization in multi GPU and cluster setup. This allows easy deployment of software within an organization. The support model has no option for commercial support, but the public support is strong due to the numerous resources being available, e. g., mailing lists and stackoverflow question boards [35]. In early releases TensorFlow was rather slow, especially for CNN network architectures [32]. This changed with recent releases and the performance is now comparable with other frameworks [33].

An additional argument in favor of TensorFlow is the support of deep learning hardware introduced by Google (tensor processing units - TPUs). When they finally will become available for the consumer market, TensorFlow-based implementations will work directly on the new hardware, allowing for another massive decrease in runtimes (the speedup is massive and supposed to be 15-30x [36]). These TPUs were used recently e. g., as infrastructure for the AlphaGo network that defeated the GO champions Fan Hui and Lee Sedol [4].

### DeepLearning4j

DeepLearning4j (DL4J) is a recent framework established mid-2015. It is developed by Skymind, a San Francisco-based startup, and released as open-source under Apache 2.0 license (commercial support available). Two features of this framework are of specific interest: the Java programming language and the built-in cluster support. Java allows development of easily deployable software applications, is well adopted in the professional software developer community, and supported by many development tools (IDEs). The intrinsic cluster support enables running on a Hadoop/Spark architecture. Hadoop is a framework that allows distributed storage and distributed processing of large data sets on computer clusters. Such clusters can consist of commodity hardware to get computing resources at low costs, and Spark uses this Hadoop infrastructure to do in-memory calculations on very large datasets [37]. This computing architecture is typical for so-called "Big-data" computing, e. g., for real-time analytics in domains where scalability is a key asset. Hadoop/Spark are also part of numerous cloud solutions where access to such infrastructures is fast and unproblematic (e. g., Amazon AWS EMR). Despite these advantages, DL4j is not the best-performing solution, as measured on single CPU environments [38], which drawback partially can be compensated when using multi GPU/CPU instances.

## Wrappers for High-Level access to Framework Functionality

A wrapper provides a high-level (and therefore often easier) access to functionality of software systems / frameworks. Wrappers are available for a few deep learning frameworks, enabling quick and easy access to functionality that is otherwise hard to access. The development of applications - for example the training of neural networks - requires writing large amounts of program code for simple, but repeated tasks. This is the main reason for using the high-level wrappers, which permit an abstract, controlled access to repeated program functions without the need of 'reinventing the wheel'. Lasagne [39] has become such a high-level lightweight framework for Theano, yet it also enables low-level Theano commands. For TensorFlow, the situation is quite diverse and also changing. There are multiple wrappers on top of TensorFlow, e. g., TFlearn. It provides a number of useful high-level features such as training data augmentation, dropout and batch normalization, and handling of hdf5 files. Both, Lasagne as well as TFlearn are community-driven third party projects that support specific build versions of the underlying frameworks only. Early 2017, Google announced to include Keras [40] with TensorFlow's core to overcome those 'synchronization' issues. Keras also remains compatible with Theano and allows mixing low- and high-level code in each framework.

## Summary

In this article we gave an overview and introduction of deep learning and frameworks that are currently available in the public domain. The criteria that we apply reflect the needs from agile software development, and are based on our own experience in developing applications during the last four years. The baseline of our comparison is the need to react fast to changed requirements and to cope with a rapid algorithmic and intellectual turnover rate in this quickly moving field. A fast adoption of new research into software frameworks is a key asset as discussed in this article.

The four key factors for a successful enablement of deep learning are a) the ease of use of such frameworks, b) the availability of stable implementations for the 'cutting-edge' algorithmic advancements in this field, c) the flexibility and possible scaling effects on different hardware architectures and d) the support available.

Given these criteria, we decided to adopt TensorFlow and its family of high-level wrappers (Keras, TFlearn), for developing deep learning applications including de-novo network design. Other frameworks, especially MXNet, might become strong alternatives in the future, especially when dedicated hardware solutions become available. However, all analyzed frameworks have their own strengths, like the easiness in switching between network models (Caffe) or specific programming language support (DeepLeaarning4J). Finally, it is advisable to continue observing the field closely, since it develops rapidly adopting algorithms and new hardware to speed up computations. This will lead eventually to new use cases and applications which are even not foreseen as of today.

## Funding

The authors received financial support for the research, authorship, and/or publication of this article, from the Commission for Technology and Innovation CTI, Switzerland.

### Conflict of Interest

The authors declared the following potential conflicts of interest with respect to the research, authorship, and/or publication of this article: Daniel Siegismund, Stephan Heyse and Stephan Steigele are employed by Genedata AG.

### References

[1] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015; 521: 436–444

[2] Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. DTIC Document 1985: 1–34

[3] LeCun Y, Bengio Y. Convolutional networks for images, speech, and time series. In: The handbook of brain theory and neural networks. 1995: 3316

[4] Silver D, Huang A, Maddison CJ et al. Mastering the game of Go with deep neural networks and tree search. Nature 2016; 529: 484–489

[5] He K, Zhang X, Ren S et al. Deep residual learning for image recognition. arXiv:1512.03385 2015

[6] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 2012; 1097–1105

[7] Mohamed A, Sainath TN, Dahl G et al. Deep belief networks using discriminative features for phone recognition. Proc Speech and Signal Processing (ICASSP) IEEE Int Conf Acoustics: 5060-5063

[8] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions 2016; arXiv:1605.02688

[9] Ekins S. The next era: Deep learning in pharmaceutical research. Pharmaceutical Research 2016; 33: 2594–2603

[10] Angermueller C, Paernamaa T, Parts L et al. Deep learning for computational biology. Molecular Systems Biology 2016; 12: 878

[11] Gonczarek A, Tomczak JM, Zaręba S et al. Interaction prediction in structure-based virtual screening using deep learning. Computers in Biology and Medicine 2017; S0010-4825: 30297–4

[12] Pereira JC, Caffarena ER, dos Santos CN. Boosting docking-based virtual screening with deep learning. Journal of Chemical Information and Modeling 2016; 56: 2495–2506

[13] Heffernan R, Paliwal K, Lyons J et al. Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. Scientific Reports 2015; 5: 11476

[14] Goh GB, Siegel C, Vishnu A et al. Chemception: A deep neural network with minimal chemistry knowledge matches the performance of expert-developed QSAR/QSPR Models. 2017; arXiv:1706.06689

[15] Mayr A, Klambauer G, Unterthiner T et al. Toxicity prediction using deep learning. Frontiers in Environmental Science 2016; 3: 80

[16] Aliper A, Plis S, Artemov A et al. Deep learning applications for predicting pharmacological properties of drugs and drug repurposing using transcriptomic data. Molecular Pharmaceutics 2016; 13: 2524–2530

[17] Chen Y, Li Y, Narayan R et al. Gene expression inference with deep learning. Bioinformatics 2016; 32: 1832–1839

[18] Angermueller C, Lee HJ, Reik W et al. DeepCpG: Accurate prediction of single-cell DNA methylation states using deep learning. Genome biology 2017; 18: 67

[19] Alipanahi B, Delong A, Weirauch MT et al. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. Nature Biotechnology 2015; 33: 831–838

[20] Dürr O, Sick B. Single-cell phenotype classification using deep convolutional neural networks. Journal of Biomolecular Screening 2016; 21: 998–1003

[21] Kraus OZ, Ba JL, Frey BJ. Classifying and segmenting microscopy images with deep multiple instance learning. Bioinformatics 2016; 32: i52–i59

[22] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks For Biomedical Image Segmentation. 2015; arXiv:150504597

[23] Sirinukunwattana K, Pluim JPW, Chen H et al. Gland segmentation in colon histology images: The GlaS challenge contest. 2016; arXiv:160300275

[24] Stern U, He R, Yang CH. Analyzing animal behavior via classifying each video frame using convolutional neural networks. Scientific Reports 2015; 5: 14351

[25] Ciresan DC, Meier U, Masci J et al. Multi-column deep neural network for traffic sign classification. Neural Networks 2012; 32: 333–338

[26] The HDF Group. Hierarchical Data Format, version 5, 1997-2017. http://www.hdfgroup.org/HDF5/

[27] Cho K, van Merrienboer B, Gülcehre C et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. 2014; arXiv:1406.1078

[28] Comparison of deep learning software. (n.d.). In Wikipedia. Retrieved June, 2017, from https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

[29] Miguel JP, Mauricio D, Rodriguez G. A Review of software quality models for the evaluation of software products. International Journal of Software Engineering & Applications 2014; 5: 31–53

[30] Adewumi A, Misra S, Omoregbe N. A review of models for evaluating quality in open source software. IERI Procedia 2013; 4: 88–92

[31] Bahrampour S, Ramakrishnan N, Schott L et al. Comparative study of deep learning software frameworks. 2015; arXiv:151106435

[32] Tran K. Deepframeworks – Evaluation of Deep Learning Frameworks GitHub repository 2017; https://github.com/zer0n/deepframeworks

[33] Chintala S. Convnet-benchmarks – Easy benchmarking of all publicly accessible implementations of convnets. GitHub repository 2017; https://github.com/soumith/convnet-benchmarks

[34] Chen T, Li M, Li Y et al. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. 2015; arXiv:151201274

[35] Abadi M, Agarwal A, Barham P et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015; arXiv:1603.04467

[36] Jouppi NP, Young C, Patil N et al. In-Datacenter performance analysis of a tensor processing unit. 44th International Symposium on Computer Architecture (ISCA) 2017: 1–12

[37] Zaharia M, Chowdhury M, Das T et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation 2012; 1–12

[38] Kovalev V, Kalinovsky A, Kovalev S. Deep Learning with Theano, Torch, Caffe, Tensorflow, and Deeplearning4J: Which one is the best in speed and accuracy? XIII Int. Conf. on Pattern Recognition and Information Processing 2016; Conference Paper pp. 99–103

[39] Dieleman S, Schlüter J, Raffel C et al. Lasagne: First release. 2015: https://zenodo.org/record/27878

[40] Chollet F. Keras. GitHub repository 2015: https://github.com/keras-team/keras