Artificial Intelligence Review (2019) 52:77–124
https://doi.org/10.1007/s10462-018-09679-z

Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey

Giang Nguyen[1] · Stefan Dlugolinsky[1] · Martin Bobák[1] · Viet Tran[1] · Álvaro López García[2] · Ignacio Heredia[2] · Peter Malík[1] · Ladislav Hluchý[1]

Published online: 19 January 2019

Abstract

The combined impact of new computing resources and techniques with an increasing avalanche of large datasets, is transforming many research areas and may lead to technological breakthroughs that can be used by billions of people. In the recent years, Machine Learning and especially its subfield Deep Learning have seen impressive advances. Techniques developed within these two fields are now able to analyze and learn from huge amounts of real world examples in a disparate formats. While the number of Machine Learning algorithms is extensive and growing, their implementations through frameworks and libraries is also extensive and growing too. The software development in this field is fast paced with a large number of open-source software coming from the academy, industry, start-ups or wider open-source communities. This survey presents a recent time-slide comprehensive overview with comparisons as well as trends in development and usage of cutting-edge Artificial Intelligence software. It also provides an overview of massive parallelism support that is capable of scaling computation effectively and efficiently in the era of Big Data.

Keywords Machine Learning · Deep Learning · Large-scale data mining · Artificial Intelligence software · Parallel processing · Intensive computing · Graphics processing unit (GPU)

Contents

1 Introduction . . . 78
2 Machine Learning and Deep Learning . . . 80
  2.1 Machine Learning process . . . 80
  2.2 Neural Networks and Deep Learning . . . 82
3 Accelerated computing . . . 85
4 Machine Learning frameworks and libraries . . . 87

This work is supported by the "Designing and Enabling E-infrastructures for intensive Processing in a Hybrid DataCloud" (DEEP-Hybrid-DataCloud) project that has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 777435.

✉ Giang Nguyen
giang.ui@savba.sk

Extended author information available on the last page of the article

4.1 Machine Learning frameworks and libraries without special hardware support 90
4.1.1 Shogun 90
4.1.2 RapidMiner 90
4.1.3 Weka3 93
4.1.4 Scikit-Learn 94
4.1.5 LibSVM 94
4.1.6 LibLinear 95
4.1.7 Vowpal Wabbit 95
4.1.8 XGBoost 96
4.1.9 Interactive data analytic and visualization tools 97
4.1.10 Other data analytic frameworks and libraries 97
4.2 Deep Learning frameworks and libraries 99
4.2.1 TensorFlow 99
4.2.2 Keras 102
4.2.3 Microsoft CNTK 103
4.2.4 Caffe 103
4.2.5 Caffe2 104
4.2.6 Torch 105
4.2.7 PyTorch 105
4.2.8 MXNet 106
4.2.9 Chainer 107
4.2.10 Theano 108
4.2.11 Performance-wise preliminary 108
4.2.12 Deep Learning wrapper libraries 109
4.3 Machine Learning and Deep Learning frameworks and libraries with MapReduce 110
4.3.1 Deeplearning4j 112
4.3.2 Apache Spark MLlib and Spark ML 112
4.3.3 H2O, Sparkling Water and Deep Water 114
4.3.4 Other frameworks and libraries with MapReduce 115
5 Conclusions 116
References 117

1 Introduction

Data mining (DM) is the core stage of the knowledge discovery process that aims to extract interesting and potentially useful information from data (Goodfellow et al. 2016; Mierswa 2017). Although during this work, the term "data mining" is primarily oriented to large-scale data mining, many techniques that perform well for large-scale datasets can also be efficiently applied to smaller datasets. Data mining can serve as a foundation for Artificial Intelligence and Machine Learning. Many techniques in this direction can be grouped in one of the following fields:

– *Artificial Intelligence* (AI) is any technique that aims to enable computers to mimic human behaviour, including machine learning, natural language processing (NLP), language synthesis, computer vision, robotics, sensor analysis, optimization and simulation.
– *Machine Learning* (ML) is a subset of AI techniques that enables computer systems to learn from previous experience (i.e. data observations) and improve their behaviour for a given task. ML techniques include Support Vector Machines (SVM), decision trees, Bayes learning, k-means clustering, association rule learning, regression, neural networks, and many more.
– *Neural Networks* (NNs) or artificial NNs are a subset of ML techniques, loosely inspired by biological neural networks. They are usually described as a collection of connected units, called artificial neurons, organized in layers.

– *Deep Learning* (DL) is a subset of NNs that makes the computational multi-layer NN feasible. Typical DL architectures are deep neural networks (DNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), generative adversarial networks (GAN), and many more.

As we explain in Sect. 2, applying one of these techniques to a given set of data can take a considerable amount of time, depending on the computing and storage capabilities that are available for scientist. In this context, different types of accelerators capable of computing specific tasks have been successfully used in many areas to complement and unburden more generic CPUs; e.g., arithmetic co-processors for floating point operations, sound cards for audio encoding/decoding or 3D graphics accelerators for image-array operations and transformations. Recently, there is a rise of new types of accelerators such as cryptographic and AI accelerators [like Google's TPUs (Jouppi et al. 2017) or Intel's Neural Processors (Kloss 2017)], which belong to new hardware architecture generation optimized for ML, DL and other AI workloads (Kobielus 2018; Tang 2018).

Generally speaking, graphics processing units (GPUs) have become the main tools for speeding up general purpose computation in the last decade (Cano 2018). They offer a massive parallelism to extend algorithms to large-scale data for a fraction of cost of a traditional high-performance CPU cluster, allowing scalability over uncomputable datasets through traditional parallel approaches.

These accelerators play a significant role in accelerating ML and DL algorithms in areas such as:

– *Image and video processing* satellites images (e.g. fires, droughts, crops diseases, urban development), space (telescope images), biology image recognition (e.g. plant, cells, bacteria), medical image recognition (e.g. magnetic resonance imaging, computer tomography, sonography, histopathology) (Torro et al. 2017; Otalora et al. 2018), computer vision (Giordaniello et al. 2017), automatic picture or audio annotations (Hafiane et al. 2017).
– *Speech and language* text processing and recognition, speech recognition, machine translation, natural language processing;
– *Security* biometrics authentication (e.g. people, faces, gait) and cyber-security (e.g. network and resource monitoring, anomaly detection, intrusion detection).
– *Business intelligence* insurance, financial markets, stock and exchange rate (e.g. time-series monitoring and predictions).
– *Robotics and video games* autonomous navigation (e.g. car, drone, plane, submarine), video games (e.g. Atari, Dota, Starcraft).

DM techniques rely on data to be analyzed in order to get insights of these data providing relevant information for the problem that is being analyzed. Nowadays, the Big Data culturermeated through all the disciplines and research areas (including computer science, medicine, finance, etc.) because of its potential within all these fields. The change in data generation and data collection has led to changes in data processing as well. The Big Data definition is characterized by many Vs, such as Volume, Velocity and Variety, as well as Veracity, Variability, Visualization, Value and so on.

At the first sight, general audience relates Big Data processing with distributed platforms like Apache Hadoop and Spark. The contiguity goes there for Volume in conformity with Veracity characteristic (Nguyen et al. 2018) as well as high-speeds in processing and inference in background. In the new era of Big Data, data analysis is expected to change. The nature of large-scale data requires new approaches and new tools that can accommodate them with different data structures, different spatial and temporal scales (Liu et al. 2016). The surge of large Volume of information, especially with the Variety characteristic, to be processed by data mining and ML algorithms demand *new transformative parallel and distributed computing solutions* capable to scale computation effectively and efficiently (Cano 2018).

In this context, this survey presents a comprehensive overview with comparisons as well as trends in development and usage of cutting-edge AI software, libraries and frameworks, which are able to learn and adapt from previous experience using ML and DL techniques to perform more accurate and more effective operations for problem solving (Rouse 2018). This survey also provides a brief overview of massive parallelism support that is capable to scale computation effectively and efficiently for large-scale datasets.

The rest of the work is organized as follows. Section 2 presents an overview of ML and DL techniques, their evolution and emerging trends. The connection between DL and accelerated computing is presented in Sect. 3. The main part of the work is Sect. 4, which provides the state-of-the-art in ML and DL frameworks and libraries. It is divided into three subsections: Machine Learning frameworks and libraries without special hardware supports (Sect. 4.1), Deep Learning frameworks and libraries with GPU support (Sect. 4.2), and Machine Learning and Deep Learning frameworks and libraries with MapReduce support (Sect. 4.3). Finally, Sect. 5 concludes the survey with a short summary about the trend in this research and developments directions.

2 Machine Learning and Deep Learning

2.1 Machine Learning process

Realization of DM in many life areas led to the Cross-Industry Standard Process for Data Mining cycle (CRISP-DM 1999), which is now the leading de facto standard for DM applications. The CRISP-DM cycle (Fig. 1) consists of six phases:

1. *The business understanding* is usually based on the provided quest formulations and data description.
2. *The data understanding* is based on the provided data and its documentation.
3. *The data preparation* consists of data transformation, exploratory data analysis (EDA) and feature engineering. Each of them can be further divided into smaller sub-steps; e.g., feature engineering consists of feature extraction, feature selection.
4. In *the modelling* phase, various ML algorithms can be applied with different parameter calibrations. The combination between data and parameter variability can lead to extensive repeating of the model train-test-evaluation cycle. If the data is large-scale, the modeling phase will have time-consuming and compute-intensive requirements.
5. *The evaluation* phase can be performed under various criteria for thorough testing of the ML models in order to choose the best model for the deployment phase.
6. *The deployment* phase, also called production phase, involves usage of a trained ML model to exploit its functionality, as well as the creation of a data pipeline into production.

The whole CRISP-DM cycle is repetitive. The group of the first five phases, called the development phase, can be repeated with different settings according to evaluation results. The deployment phase is critical for real production under repetitive requisites; it implies online evaluation, monitoring, model maintenance, diagnosis, and retraining. It is needed to highlight that ML algorithms learn from data. Therefore, in practice, it is expected that the data understanding and data preparation phases can consume a large portion of the entire time of every DM project.

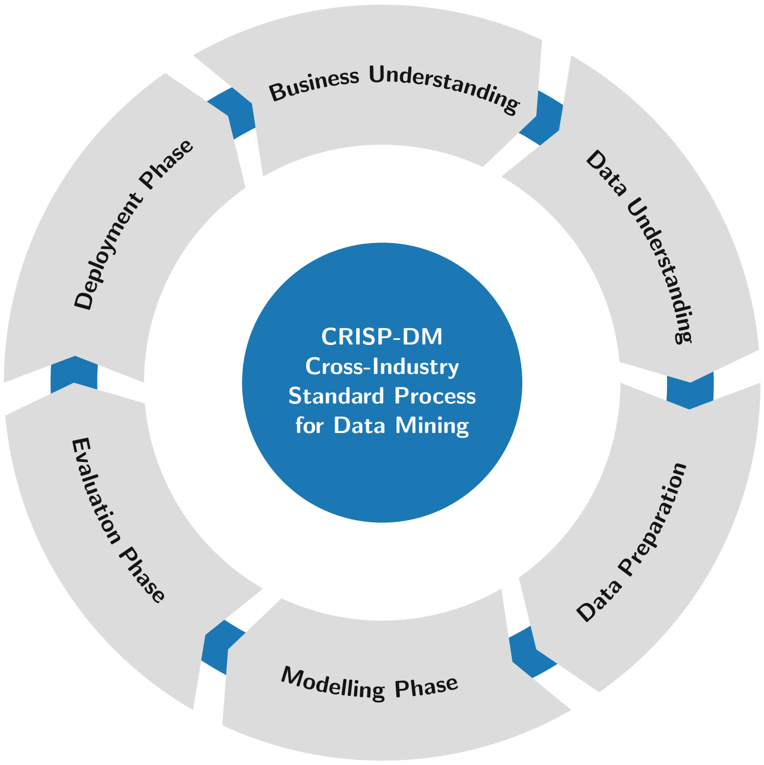

Fig. 1 CRISP-DM cross-industry standard process for data mining

Data preparation methods (including data transformation, EDA and feature engineering) can be categorized into a number of sub-categories such as dimensionality reduction, sampling (sub-sampling, oversampling), data augmentation, linear methods, statistical testing, feature engineering with feature extraction, feature encoding, feature transformation and feature selection (e.g. mutual information, chi-square χ^2 statistics, sensitivity analysis) (Bachniak et al. 2017). There are also many other algorithms for over-fitting prevention; e.g., regularization, threshold setting, pruning, or dropout.

Once the data is prepared, one has to choose a ML algorithm to solve the problem. When facing a huge number of different ML algorithms, the most frequent question is: "Which algorithm is the right solution for the given problem?". The answer to this question varies depending on many factors, including

1. size, quality, and nature of the domain data;
2. available computational time;
3. urgency of the task and
4. structure of the desired predictions and loss to be minimized.

Algorithms can be grouped according to the task they intend to solve. Below is one of the widely accepted ML algorithm categorizations (Bishop 2006):

– *Supervised learning* algorithms are used when each example in the training data consists of a pair (X_i, y_i), where X_i is the input to be fed into the predictor and y_i is the ground-truth label of the input. The training routine consists of tuning the predictor's parameters to make its outputs $f(X_i)$ as close to y_i as possible. Most of the classification and regression problems fall under this umbrella.

- *Unsupervised learning* is used to extract information from training data where a ground-truth label y_i is not available. Among the techniques that would belong to this category is clustering, density estimation, dimensionality reduction, generative adversarial networks or problems where $X_i = y_i$ such as autoencoders.
- *Reinforcement learning* is used to design algorithms that select appropriate actions in a given situation in order to maximize reward. In contrast to supervised learning where the predictor is provided with ground-truth labels, here the algorithm must learn by trial and error to find the optimal outputs.

It is interesting to note that the number of ML algorithms is increasing continually and ML algorithms have no strict categorization. It means that some methods can be listed in more categories. For example, NNs can be used to solve supervised and reinforcement learning problems.

Often it is difficult to predict which algorithm would perform the best before trying different algorithms after a thoughtful data examination. The choice of a concrete algorithm is based on data characteristics and EDA. As it is usual in ML, the performance of data models strongly depends on the representativeness of the provided data set. The complementarity of methods leads to a selection from a wide spectrum of available modelling methods based on data characteristics and analysis.

In order to reach the maximum performance, in many cases, it is necessary to train each model multiple times with different parameters and options (so-called model ensembling). Sometimes, it is also suitable to combine several independent models of different types, because each type can be strong in fitting different cases. The full data potential can be reached by ensemble learning based on principles such as voting, record weighting, multiple training process or random selection. Weak learners can also be converted into stronger ones using boosting methods. Hence, a proper combination of several types of models with different advantages and disadvantages can be used to reach the maximum accuracy and stability in predictions.

Model selection and model performance optimization contains a large number of approaches such as hyper-parameter tuning, grid search, bayesian optimization (Snoek et al. 2012), local minimum search, and bio-inspired optimization; e.g., evolutionary algorithms (Triguero et al. 2017; Alcala-Fdez et al. 2011) and genetic programming (Cano et al. 2015). There are also many methods that can be applied for model evaluation, such as cross-validation, k-fold, holdout with various metrics such as accuracy (ACC), precision, recall, F1, Matthews correlation coefficient (MCC), receiver operating characteristic (ROC), area under the curve (AUC), mean absolute error (MAE), mean squared error (MSE), and root-mean-square error (RMSE).

2.2 Neural Networks and Deep Learning

As previously described, *Neural Networks* (NNs) are a subset of ML techniques. These networks are not intended to be realistic models of the brain, but rather robust algorithms and data structures able to model difficult problems. NNs have units (neurons) organized in layers. There are basically three layer categories: input layers, hidden (middle) layers and output layers. NNs can be divided into shallow (one hidden layer) and deep (more hidden layers) networks. The predictive capability of NNs comes from this hierarchical multilayered structure. Through proper training the network can learn how to optimally represent inputs as features at different scales or resolutions and combine them into higher-order feature representations relating these representations to output variables and therefore learning how to make predictions.

Deep Neural Networks (DNNs) are considered to be capable of learning high-level features with more complexity and abstraction due to their larger number of hidden layers than shallow NNs. In order to achieve high accuracy in prediction, there are two dependent problems that have to be addressed when solving problems with NNs; i.e., network architecture and training routine (Goodfellow et al. 2016; LISA 2015; Schmidhuber 2015):

– Defining network architectures involves setting fine-grained details such as activation functions [e.g. hyperbolic tangent, rectified linear unit (ReLU), maxout] and the types of layers (e.g. fully connected, dropout, batch normalization, convolutional, pooling) as well as the overall architecture of the network.
– Defining training routines involves setting the learning rate schedules (e.g. stepwise, exponential), the learning rules [e.g. stochastic gradient descent (SGD), SGD with momentum, root mean square propagation (RMSprop), Adam], the loss functions (e.g. MSE, categorical cross entropy), regularization techniques (e.g. L1/L2 weights decay, early stopping) and hyper-parameter optimization (e.g. grid search, random search, bayesian guided search).

Although NNs were proposed in the 1940s and DNNs in 1960s, the first practical application employing multiple digital neurons appeared in 1990 with the LeNet network for handwritten digit recognition. The DL successes from the 2010s are believed to be under the confluence of three main factors:

1. new algorithmic advances that have improved application accuracy significantly and broadened applicable domains;
2. availability of huge amount of data to train NNs;
3. increase of computing power.

Many DNN models have been developed over the past two decades (Deshpande 2017; Kalray 2017; Sze et al. 2017). Each of these models has a different network architecture in terms of number of layers, layer types, layer shapes and connections between layers. Table 1 presents a timeline of iconic computer vision models over the recent years with their performance in a well-known computer vision challenge, the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al. 2015).

An overview of the most DL architectures can be found in Veen (2016). In this part, we introduce some applications of the most popular architectures. Bear in mind that many applications are simultaneously composed of several types of architectures.

– *Fully Connected Networks* (*FC*) are the most common architecture as they can be used to model a wide variety of problems that use tabular data. It has been a long time since they are used in finance (Trippi and Turban 1992), physics (Duch and Diercksen 1994) or biomedicine (Wainberg et al. 2018).
– *Convolutional Neural Networks* (*CNN*) are networks specially designed to deal with images (or more generally with translation invariant data). Current applications include a wide variety of image classifiers [including wildlife (Van Horn et al. 2018), X-ray scans (Rajpurkar et al. 2017) or galaxies (Dieleman et al. 2015)], image segmentation [with applications to crowd counting (Sam et al. 2017), autonomous driving (Siam et al. 2017) or satellite imagery (Mnih 2013)], along with many other miscellaneous applications

Table 1 DL timeline through the most well-known models at ILSVRC

Year	Model	Number of layers	Top 5 error (%)
1990	LeNet	4	–
	LeNet is one of the first commercial successful CNN applications (LeCun et al. 1998). It was deployed in ATMs to recognize digits for check deposits. The most well known version is the LeNet-5		
2012	AlexNet	8	16.4
	AlexNet is the first CNN winned ILSVRC with GPUs to train the network.		
2013	ZF Net	8	11.7
	ZF Net is one of the top entries at ILSVRC 2013 (Zeiler and Fergus 2014). The architecture is very similar to AlexNet with minor modifications in the architecture hyperparameters		
2014	VGG Net	19	7.3
	VGG Net, which has the VGG-16 and VGG-19 versions, classified second in the ILSVRC 2014 (Simonyan and Zisserman 2014). The main contribution of the architecture is that the number of filters doubles after each maxpool layer. This reinforces the idea of shrinking spatial dimensions, but growing depth		
2015	GoogLeNet	22	6.7
	GoogLeNet (also referred to as the Inception) won the ILSVRC 2014 (Szegedy et al. 2015). It introduced an inception module composed of parallel connections which drastically reduced the number of parameters. From this point, CNN architectures became more than only sequential stacks. Today, GoogLeNet has 4 versions with deeper architecture (at least 42 layers) and many improvements		
2015	ResNet	152	3.57
	ResNet (also known as Residual Net) won the ILSVRC 2015 (He et al. 2016a, b), being the first network to surpass human-level accuracy with a top-5 error rate below 5%. It uses residual connections i.e. a shortcut module or bypass that allow a better gradient backward flow. ResNets have been used as a starting point to further develop new architectures, like Wide ResNets (Zagoruyko and Komodakis 2016) or DenseNets (Huang et al. 2017)		
2016	SqueezeNet	14	14.0
	SqueezeNet focuses in heavily reducing model size using deep compression (Iandola et al. 2016) without losing accuracy. The result is a network with roughly the same classification accuracy as AlexNet but with 510 times less memory requirements (0.5 MB of memory when tested on the ILSVRC 2012 dataset)		

involving images such as poverty prediction from satellite imagery (Jean et al. 2016). Recently, CNNs are beginning to be reconsidered as a good alternative to recurrent networks when using sequential data (Fawaz et al. 2018).

– *Recurrent Neural Networks* (*RNN*) are specially designed to deal with sequential data. They are widely used in Natural Language Processing (NLP) like Neural Machine Translation (Wu et al. 2016b), language generation (Young et al. 2018), time series analysis (Fawaz et al. 2018) [with applications in many fields such as physics (Wielgosz et al. 2017), medicine (Su et al. 2017) or climatology (Alemany et al. 2018)]. If the sequential data are images (like in videos) RNNs are used in conjunction with CNNs. In this case possible applications include video classification (Karpathy et al. 2014) or system monitoring (Pol et al. 2018).
– *Generative Adversarial Networks* (*GAN*) are networks that compete against themselves to create the most possible realistic data. Current applications include image style transfer (Zhu et al. 2017), high resolution image synthesis (Wang et al. 2018), text-to-image synthesis (Zhang et al. 2017), image super-resolution (Ledig et al. 2017) [for instance low-

dose PET reconstruction (Xu et al. 2017)], anomaly detection (Schlegl et al. 2017), 3D object generation (Wu et al. 2016a) [for instance dental restoration (Hwang et al. 2018)], music generation (Yang et al. 2017), scientific simulations acceleration [for example in astrophysics (Mustafa et al. 2017) or high energy physics (Paganini et al. 2018)] and many more.

3 Accelerated computing

DL makes profit of using a form of specialized hardware present in accelerated computing environments. The current mainstream solution (NVIDIA 2018) has been to use *Graphics Processing Unit* (GPU) as general purpose processors. GPUs provide massive parallelism for large-scale DM problems, allowing scaling algorithms vertically to data of volumes that are not computable by traditional approaches (Cano 2018). GPUs are effective solutions for real-world and real-time systems requiring very fast decision and learning, such as DL (especially in image processing).

Popular alternatives to GPUs include *Field Programmable Gate Array* (FPGA) (Lacey et al. 2016) and the recently announced Google Tensor Processing Unit 3.0 (TPU) (GoogleTPU 2018). Other IT companies are starting to offer dedicated hardware for DL acceleration; e.g., Kalray with their second generation of DL acceleration device MPAA2-256 Bostan, oriented to mobile devices such as autonomous cars (Kalray 2017). The list of DL accelerator technology providers is quite long (Tang 2018) and includes worldwide IT companies such as IBM TrueNorth Neuromorphic chip (Feldman 2016), Microsoft BrainWave (Microsoft 2017), Intel Nervana Neural Network Processor (Kloss 2017), AMD Radeon Instinct (AMD 2018), along with many startups.

This hardware has to be adapted also to the new trends that emerge in the ML/DL software development community. For example, several schemes that will greatly benefit from specialized hardware have been devised for improving the speed and memory consumption of DL algorithms; e.g.,:

– *Sparse computation*, which imposes the use of sparse representations along the neural network. Benefits include lower memory requirements and faster computation.
– *Low precision* data types (Konsor 2012), smaller than 32-bits (e.g. half-precision or integer) with experimentation even with 1-bit computation (Courbariaux et al. 2016). Again this speeds up algebra calculation as well as greatly decreasing memory consumption at the cost of a slightly less accurate model (Iandola et al. 2016; Markidis et al. 2018). In these recent years, most DL frameworks are starting to support 16-bit and 8-bit computation (Harris 2016; Andres et al. 2018).

Vertical scalability of large-scale DL is still limited due to the GPU memory capacity, which is up to 32GiB on the NVIDIA Volta architecture at the moment. Integration of MapReduce frameworks with GPU computing may overcome many of the performance limitations and it is an open challenge for future research (Cano 2018). Multi-GPU and distributed-GPU solutions are used to combine hardware resources to scale-out to bigger data (data parallelism) or bigger models (model parallelism).

Data parallelism and model parallelism are different ways of distributing an algorithm (Hermans 2017). These terms are often used in the context of ML on how to use available computing power to speed-up computations.

– Data parallelism involves the use of different nodes to run the same portion of code on different batches of data.

– Model parallelism involves the development of more sophisticated models that distribute the computation of different model subparts among different worker nodes. Currently, the limiting factor of the parallelization at the model level is the communication latency between cluster nodes, a factor that could be alleviated by the usage of specialized interconnections.

The advantages (and disadvantages) of combining of data parallelism (in the convolutional layers) and model parallelism (in the dense layers) of CNN can be found in Krizhevsky (2014).

The main feature of many-core accelerators such as GPU is their massively parallel architecture allowing them to speed up computations that involve matrix-based operations, which are at heart of many ML/DL implementations. Manufacturers often offer the possibility to enhance hardware configuration with many-core accelerators to improve machine/cluster performance as well as *accelerated libraries*, which provide highly optimized primitives, algorithms and functions to access the massively parallel power of GPUs. The most frequently mentioned accelerated libraries are:

– The NVIDIA CUDA (Compute Unified Device Architecture) (CUDA 2018) is a parallel computing platform and programming model developed by NVIDIA for general computing on GPUs. GPU-accelerated CUDA libraries enable drop-in acceleration across multiple domains such as linear algebra, image and video processing, DL and graph analytics. The NVIDIA CUDA Toolkit (CudaToolkit 2018) provides a development environment for creating high performance GPU-accelerated applications.
– The NVIDIA CUDA Deep Neural Network library (cuDNN) (cuDNN 2018), which is a GPU-accelerated library of DNN's primitives. The cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. It allows DL users to focus on training NNs and developing software applications rather than spending time on low-level GPU performance tuning. The cuDNN is used by many DL frameworks including Caffe2, CNTK, TensorFlow, Theano, PyTorch and MatLab.
– Intel MKL (Intel Math Kernel Library) (MKL 2018) optimizes code with minimal effort for future generations of Intel processors. It is compatible with many compilers, languages, operating systems, and linking and threading models. It accelerates math processing routines, increases application performance, and reduces development time. This ready-to-use core math library includes: linear algebra (BLAS/OpenBLAS, LAPACK, ScaLAPACK), sparse solvers, Fast Fourier Transforms (FFT), DNNs, vector statistics and data fitting, vector math and miscellaneous solvers.
– OpenCL (Open Computing Language) provides compatibility across heterogeneous hardware from any vendor (Khronos 2018).

As a potential forward lane, we mention the AMD ROCm (Radeon Open Compute platforM) open ecosystem. It is the AMD's answer to the widespread NVIDIA CUDA and consists of a programming-language independent open-source HPC/hyperscale-class platform for GPU computing. It features:

– peer-to-peer multi-GPU operation with Remote Direct Memory Access (RDMA) support.
– the Heterogeneous System Architecture (HSA) runtime API which provides a rich foundation to execute HCC, C++, HIP, OpenCL, and Python (ROCm 2016).
– the Heterogeneous-compute Interface for Portability (HIP) which allows developers to convert CUDA code to common C++. HIP is very thin and has little or no performance impact over coding directly in CUDA or HCC. Because both CUDA and HIP are C++

languages, porting from CUDA to HIP is significantly easier than porting from CUDA to OpenCL (ROCm-HIP 2018).
- MIOpen, which is AMD's open-source GPU-accelerated library for high performance ML primitives with large parts of the source code compatible with cuDNN (MIOpen 2018). Currently only TensorFlow, Caffe, and Caffe2 are supported by MIOpen, while other DL libraries as PyTorch, MXNet, CNTK and HIPnn are in the development list (ROCm-DL 2018).

Other programming libraries, which support computational speed-up and parallel computing, are:

- OpenMP is an Application Programming Interface (API) that supports multi-platform shared memory multiprocessing programming (OpenMP 2018). It consists of a set of compiler directives, library routines, and environment variables that influence run-time behaviour.
- Open MPI is an open-source implementation of the MPI specifications (OpenMPI 2018). The Open MPI software achieves high performance and it is quite receptive to community input. MPI stands for the Message Passing Interface—a standardised API typically used for parallel and/or distributed computing. It is written by the MPI Forum, which is a large committee comprising of a cross-section between industry and research representatives.

Application built with the hybrid model of parallel programming can run on a computer cluster using both OpenMP and MPI, such that OpenMP is used for parallelism within a (multi-core) node while MPI is used for parallelism between nodes.

4 Machine Learning frameworks and libraries

The number of ML algorithms, as well as their different software implementations, is quite large. Many software tools for DM using ML techniques have been in development for the past 25 years (Jovic et al. 2014). Their common goal is to facilitate the complicated data analysis process and to propose integrated environments on top of standard programming languages. These tools are designed for various purposes: as analytics platforms, predictive systems, recommender systems, processors (for images, sound or language). A number of them are oriented to fast processing and streaming of large-scale data, while others are specialized in implementing ML algorithms including NNs and DL. Again, it is important to emphasize that *there is no single tool suitable for every problem* and often a combination of them is needed to succeed. Figure 2 provides a comprehensive grouped overview of ML frameworks and libraries.

It is a fact that the code of many *open-source* tools is located on GitHub in the form of repositories (GitHub 2018). GitHub itself keeps a lot of monitoring information about software development such as number of contributors and commits (with historical and current activity of each team member and the team and the project as the whole), number of watches, stars, forks and issues that come with diagrams and insights. There is also a number of third-party applications connected to GitHub that provide automated code reviews and code analytics. The quality and quantity of their analytics are based on GitHub data, but differ in viewpoints, presentations, evaluation details and preference settings. Values of the popularity column of the Tables 3, 4 and 5 are estimated based on information about open-source tools in GitHub, generated analyses [from Codacy (Codacy 2018) and CodeFactor (CodeFactor 2018)] and Github star history [from CodeTabs (Jolav 2018)]. Table 2 presents a detailed snapshot (September 2018) of these results. These measures are subject to change

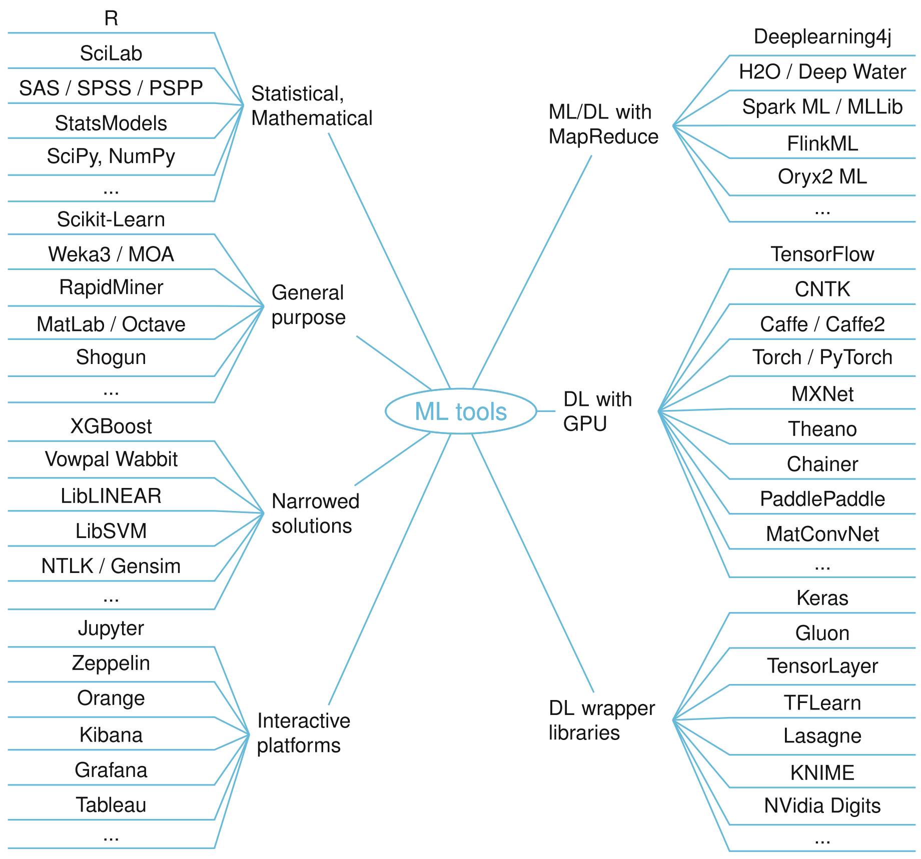

Fig. 2 Overview of Machine Learning frameworks and libraries

over time due to the hectic development of ML and DL frameworks and tools. Values of the usage column are estimated based on public information about open-source software such as official and partner websites, product presentations, success stories and publications with references to the corresponding parts.

Regarding Sects. 4.1, 4.2 and 4.3, the most well-known tools are described and evaluated briefly with their basic properties such as implementation language, license, coverage of ML methods as well as supports for recent advanced DM topics; i.e., the current demand of processing large-scale data. Most of the modern DM tools are based on dataflow architectures (pipeline or workflow). Some of them have integrated graphical user interface (GUI), others prefer an API approach or both.

Section 4.1 describes the state-of-the-art of ML/DL frameworks and libraries which do not require special hardware or infrastructure. Nevertheless, these tools can take advantage of multi-CPU computation to deal with large-scale data. Section 4.2 is devoted to DL frameworks and libraries with GPU support while Sect. 4.3 presents ML/DL frameworks and libraries integrated with Map-Reduce.

A summary of the Sects. 4.1, 4.2 and 4.3 is presented in Tables 3, 4 and 5 respectively. These tables summarize framework and library capabilities so users can choose appropriate products for tackling their problems. Each tool is also described and evaluated separately below the tables in more detail.

Table 2 Composed snapshot done based on from Github data and analysis from Codacy, CodeFactor and CodeTabs for Github star history (September 2018)

	GitHub					Codacy Code quality on master branch		CodeFactor Code quality on master branch last year		CodeTabs Github star history
	Contributors	Commits	Watch	Star	Fork	Project certification [A–D]	Issues	Code quality [A–F]	Issues	Growth speed (based on trend diagrams)
ML frameworks without special hardware support										
Shogun	147	16,763	209	2147	866	A	1021 (1%)	B+	3160	Low
Scikit-learn	1160	23,235	2188	30,536	15,076	B	1735 (10%)	B	2721	Very fast
LibSVM	13	1032	289	2824	1236	B	235 (13%)	F	247	Low
LibLinear	15	213	70	660	270	A	66 (5%)	F	71	Low
Vowpal Wabbit	151	8221	424	5809	1455	B	1429 (11%)	B	1259	Low
XGBoost	317	3438	925	13,443	5917	B	1105 (19%)	B+	132	Fast
DL frameworks with GPU support										
TensorFLow	1642	40,500	8335	109,535	67,531	B	94,272 (11%)	A−	9625	Very fast
Keras	719	4766	1849	33,538	12,645	B	1113 (22%)	D−	730	Very fast
CNTK	189	15,949	1367	15,104	4026	A	3057 (5%)	C	3611	Fast
Caffe	270	4152	2215	2552	15,614	A	184 (1%)	C+	531	Fast
Caffe2	193	3678	576	8275	2089	A	1812 (5%)	B	2721	Fast
Torch	132	1336	681	6035	2305	A	0 (0%)	C	79	Low
PyTorch	760	13,402	950	18,684	4471	A	4351 (5%)	A−	4265	Very fast
MXNet	587	7830	1170	15,197	5497	B	14,476 (17%)	B	2826	Fast
Chainer	182	15,918	324	4118	1087	B	1426 (15%)	B−	1225	Low
Theano	328	28,030	585	8477	2447	B	2901 (16%)	F	3107	Low
ML/DL frameworks with MapReduce										
Deeplearning4J	231	23,492	828	9600	4435	A	12,283 (8%)	B	10,762	Fast
Apache Spark	1286	22,729	2072	18,781	16,912	A	10,633 (7%)	A−	4755	Fast
H2O-3	109	23,246	365	3407	1309	C	14,993 (36%)	C	8510	Fast
Knime-core	32	15,426	21	93	21	B	9737 (11%)	A−	3233	Low

4.1 Machine Learning frameworks and libraries without special hardware support

4.1.1 Shogun

Shogun is a long time developed open-source general purpose ML library that offers a wide range of efficient and unified ML methods (Shogun 2018; ShogunGoogle 2018; Sonnenburg et al. 2010) built on an architecture written in C++ and licensed under GNU GPLv3 license. It has been under active development since 1999. Currently, Shogun is developed by a team of diverse volunteers and it is a sponsored project of NumFOCUS since 2017. The main idea behind Shogun is that the underlying algorithms are transparent, accessible and that anyone should be able to use them for free.

The library SVM contains 15 implementations in combination with more than 35 kernel implementations, which can be furthermore combined/constructed by sub-kernel weighting. Shogun also covers wide range of regression and classification methods as well as a number of linear methods, algorithms to train hidden Markov models, statistical testing, clustering, distance counting, FFNNs and model evaluations and many more. It has been successfully used in speech and handwriting recognition, medical diagnosis, bioinformatics, computer vision, object recognition, stock market analysis, network security, intrusion detection, and many more.

Shogun can be used transparently in many languages and environments such as Python, Octave, R, Java/Scala, Lua, C#, and Ruby. It offers bindings to other sophisticated libraries including, LibSVM/LibLinear, SVMLight, LibOCAS, libqp, Vowpal Wabbit, Tapkee, SLEP, GPML and with future plans of interfacing TensorFlow and Stan.

Strong points

- Breath-oriented ML/DM toolbox with a lot of standard and cutting-edge ML algorithms.
- Open-source, cross-platform, API-oriented, the oldest and still maintained library with core implementation in C++.
- Bindings to many other ML libraries, programming interface in many languages.

Weak points

- The most of the code has been written by researchers for their studies for a long time and therefore its code is not easily maintainable or extendable.
- Lack of documentation, suitable mainly for academic use.

The latest version of Shogun is 6.1.3 (December 2017).

4.1.2 RapidMiner

RapidMiner is a general purpose data science software platform for data preparation, ML, DL, text mining, and predictive analytics (Mierswa et al. 2003; Rapid 2018). RapidMiner (formerly YALE, Yet Another Learning Environment) was developed starting in 2001 at the Artificial Intelligence Unit of the Technical University of Dortmund.

It is a cross-platform framework developed on open core model written in Java. RapidMiner supports interactive mode (GUI), command-line interface (CLI) and Java API. It is mainly a proprietary commercial product since version 6.0. Its architecture is based on a client/server model with server offered as either on-premise, or in public or private cloud infrastructures (Amazon AWS, and Microsoft Azure). For large-scale data analytics, RapidMiner supports unsupervised learning in Hadoop (Radoop), supervised learning in memory

Table 3 ML frameworks and libraries without special hardware supports

Tool	Licence	Written in	Algorithm coverage	Interface	Workflow	Popularity	Usage	Creator (note)
Shogun (ML library)	Open source, GNU GPLv3	C++	High	Python, Octave, R, Java/Scala, Lua, C#, Ruby	API	Low	Academic	G. Raetsch, S. Sonnenburg **NUMFOCUS**
RapidMiner[a] (ML/NN/DL framework)	Business source	Java	High	Python, R, GUI, API	Yes	High	Academic	R. Klinkenber, I. Mierswa, S. Fischer., et al **RapidMiner**
Weka[b] (ML/DL framework)	Open source, GNU GPLv3	Java	High	Java, GUI, API	Yes	High	Academic	**University of Waikato**, New Zealand
Scikit-Learn (ML/NN library)	Open source, BSD	Python, C++	High	Python, API	Yes	High	Academic	D. Cournapeau **INRIA, Google and others**
LibSVM (ML library)	Open source, BSD 3-clause	C/C++	Low (only SVM)	Python, R, MatLab, Perl, Ruby, Weka, Lisp, Haskell, OCaml, LabView, PHP . . .	No	Low	Academic	C.C. Chang, C.J. Lin **Taiwan National University**
LibLinear (ML library)	Open source, BSD 3-clause	C/C++	Low (only linear)	MatLab, Octave, Java, Python, Ruby . . .	No	Low	Academic Industrial	R.E.Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, C.J. Lin **Taiwan National University**

Table 3 continued

Tool	Licence	Written in	Algorithm coverage	Interface	Workflow	Popularity	Usage	Creator (note)
Vowpal Wabbit (ML library)	Open source, BSD 3-clause	C++, own MPI library AllReduce	Low	API	No	Medium	Academic Industrial	J. Langford **Microsoft**, previously Yahoo
XGBoost (ML boosting, ensemble)	Open source, Apache 2.0	C++	Low	C++, Java, Python, R, Julia	Yes	Medium	Academic Industrial	T. Chen **DLMC group**

[a] Although the core of RapidMiner stays open-source, RapidMiner changed its model to business source (RapidMiner 2013)
[b] Weka uses subversion repository (SVN) and it has a read-only repository on GitHub for stable releases (Waikato 2018)

with scoring on the cluster (SparkRM) and scoring with native algorithms on the cluster. In this case, the algorithm coverage is narrowed into Naive Bayes, linear regression, logistic regression, SVM, decision tree, random forest and clustering using k-means and fuzzy k-means.

Although the core of RapidMiner stays open-source, RapidMiner changes its model to *business source*, that means the latest version will be available as a trial version or under an commercial license (RapidMiner 2013). The free edition, available under the AGPL license, is limited to one logical processor and 10,000 data rows.

Strong points

- General purpose, wide set of algorithms with learning schemes, models and algorithms from Weka and R scripts.
- Add-ons support with selected algorithms for large-scale data.
- Strong community, cross-platform framework.

Weak points

- Proprietary product for larger problem solutions.

The latest version of RapidMiner is 9.0 (August 2018).

4.1.3 Weka3

Weka collects a general purpose and very popular wide set of ML algorithms implemented in Java and engineered specifically for DM (Weka3 2018; Waikato 2018) . It is a product of the University of Waikato, New Zealand and is released under GNU GPLv3-licensed for non-commercial purposes.

Weka has a package system to extend its functionality, with both official and unofficial packages available, which increases the number of implemented DM methods. It offers four options for DM: command-line interface (CLI), Explorer, Experimenter, and Knowledge Flow.

Weka can be used with Hadoop thanks to a set of wrappers produced for the most recent versions of Weka3. At the moment, it supports MapReduce but not yet Apache Spark. Clojure (Hickey 2018) users can also leverage Weka, thanks to the Clj-ml library (Clj-ml 2018). Related to Weka, Massive Online Analysis is also a popular open-source framework written in Java for data stream mining, while scaling to more demanding larger-scale problems.

Strong points

- General purpose, involving wide set of algorithms with learning schemes, models and algorithms.
- It comes with GUI and is API-oriented.
- Supports standard DM tasks, including feature selection, clustering, classification, regression and visualization.
- Very popular ML tool in the academic community.

Weak points

- Limited to Big Data, text mining, and semi-supervised learning.
- Weak for sequence modelling; e.g., time-series.

The latest stable version of Weka is 3.8.3 (September 2018).

4.1.4 Scikit-Learn

Scikit-Learn is widely known as a popular open-source Python tool which contains comprehensive library of DM/ML algorithms (Scikit 2018). The Scikit-Learn project started as a Google Summer of Code project by David Cournapeau. Since 2015, it is under active development sponsored by INRIA, Telecom ParisTech and occasionally Google through the Google Summer of Code.

It extends the functionality of NumPy and SciPy packages with numerous DM algorithms and provides functions to perform classification, regression, clustering, dimensionality reduction, model selection and preprocessing. It also uses the Matplotlib package for plotting charts.

Since April 2016, Scikit-Learn is provided in jointly-developed Anaconda (Anaconda 2018) for Cloudera project on Hadoop clusters (AnacondaCloudera 2016). In addition to Scikit-Learn, Anaconda includes a number of popular packages for mathematics, science, and engineering for the Python ecosystem such as NumPy, SciPy and Pandas.

Strong points

- General purpose, open-source, commercially usable, and popular Python ML tools.
- Funded by INRIA, Telecom Paristech, Google and others.
- Well-updated and comprehensive set of algorithms and implementations.
- It is a part of many ecosystems; it is closely coupled with statistic and scientific Python packages.

Weak points

- API-oriented only.
- The library does not support GPUs.
- Basic tools for NNs.

The latest released version of Scikit-Learn is 0.20.0 (September 2018), which came with declaration of dropping support for Python 3.4 and below in the incoming version 0.21.0.

4.1.5 LibSVM

LibSVM is a specialized library for Support Vector Machines (SVM). Its development started in 2000 at National Taiwan University (Chang and Lin 2011; LibSVM 2018).

The library is written in C/C++ but has also Java source code. Its learning tasks are (1) support vector classification (SVC) for binary and multi-class, (2) support vector regression (SVR), and (3) distribution estimation. Supported problem formulation are: C-SVC, υ-SVC, distribution estimation (one-class SVM), ε-SVR, and υ-SVR. All of the formulations are quadratic minimization problems and are solved by sequential minimal optimization algorithm. The running time of minimizing SVM quadratic problems is reduced by shrinking and caching. LibSVM provides some special settings for unbalanced data by using different penalty parameters in the SVM problem formulation. It has been successfully used in computer vision, NLP, neuro-imaging, and bioinformatics with 250K downloads in the 2000–2010 period.

LibSVM provides interfaces for Python, R, MATLAB, Perl, Ruby, Weka, Common LISP, CLISP, Haskell, OCaml, LabVIEW, and PHP. Its code is also reused in DM tools like Weka, RapidMiner, and KNIME. Scikit-Learn declares to use LibSVM to handle computations internally but with modifications and improvements. The library is popular in the open-source ML community and is released under the 3-clause BSD license.

Strong points

– The LibSVM data format is a specific data format for the data analysis tool LibSVM, which is well-accepted in other frameworks and libraries. The format is dense and suitable to describe and process Big Data especially because it allows for a sparse representation.
– Open-source, specialized tool with high popularity in the open-source ML community.

Weak points

– LibSVM training algorithm does not scale up well for very large datasets in comparison to LibLinear or Vowpal Wabbit (Zygmunt 2014). It takes $O(n^3)$ time (Abdiansah and Wardoyo 2015) in the worst case and around $O(n^2)$ on typical cases, where n is the number of data points.
– Limited to problems where SVM performs well.

The latest version of LibSVM is 3.23 (June 2018), which fixes minor issues.

4.1.6 LibLinear

LibLinear is a library designed for solving large-scale linear classification problems. It was developed starting in 2007 at National Taiwan University (Fan et al. 2008; LibLinear 2018).

The library is written in C/C++. The supported ML tasks are logistic regression and linear SVM. The supported problem formulation are: $L2$-regularized logistic regression, $L2$-loss and $L1$-loss linear SVMs. The approach for $L1$-SVM and $L2$-SVM is a coordinate descent method. For linear regression (LR) and $L2$-SVM, LibLinear implements a trust region Newton method. For multi-class problems, LibLinear implements the one-vs-rest strategy and the Crammer & Singer method (Crammer and Singer 2001, 2002).

LibLinear provides interfaces for MatLab, Octave, Java, Python and Ruby. Its code is also reused in DM tools like Weka and KNIME. Scikit-Learn declares to use LibLinear to handle computations internally but with modifications and improvements. The ML group at National Taiwan University also provides support (in early stages) for MPI LibLinear, which is an extension of LibLinear for distributed environments and for Spark LibLinear, which is Spark implementation based on LibLinear and integrated with Hadoop distributed file system (NTU 2018). The library is popular in the open-source ML community and it is released under the 3-clause BSD license.

Strong points

– Designed to solve large-scale linear classification problems.
– Open-source, specialized tool with high popularity in open-source ML community.

Weak points

– Limited to LR and linear SVM.

The latest version of LibLinear is 2.20 (December 2017).

4.1.7 Vowpal Wabbit

Vowpal Wabbit (or VW) is an efficient scalable implementation of online ML and supports various incremental ML methods (VW 2018; VWAzure 2018). It is an open-source fast out-of-core learning system originally developed by John Langford at Yahoo! Research, and is currently sponsored by Microsoft Research.

VW is one of the offered ML options in Microsoft Azure. Its many features include reduction functions, importance weighting, selection of different loss functions and optimization algorithms. VW has been used to learn a tera-feature (10^{12}) data-set on thousand (10^3) nodes in 1 h, and can run properly in single machine, Hadoop and HPC cluster.

Strong points

– Open-source, efficient, scalable and fast out-of-core online learning supported by strong IT companies (Microsoft, previously Yahoo).
– Feature identities are converted to a weight index via a hash using 32-bit MurmurHash3. Feature hashing, or the hashing trick (Weinberger et al. 2009) is a fast and space-efficient way of vectorizing features that enables online learning at speed.
– Exploiting multi-core CPUs on Hadoop cluster by own MPI-AllReduce library, parsing of input and learning are done in separate threads.
– Allows using non-linear features e.g., n-grams.
– Product a the strong industrial laboratory, compiled C++ code.
– One of the offered ML options in Microsoft Azure.

Weak points

– The number of available ML methods is sufficient but limited.

The latest release of Vowpal Wabbit is the version 8.6.1 (July 2018).

4.1.8 XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable (Chen and Guestrin 2016; DMLC 2018; Mitchell 2017).

The XGBoost is an open-source library that implements the gradient boosting decision tree algorithm. It has gained much popularity and attention recently as it was the algorithm of choice for many winning teams of a number of ML competitions. XGBoost implements ML algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT or GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples. The term gradient boosting comes from the idea of boosting or improving a single weak model by combining it with a number of other weak models in order to generate a collectively strong model. XGBoost boosts the weak learning models by iterative learning.

It provides interfaces for C++, Java, Python, R, and Julia and works on Linux, Windows, and Mac OS. It also supports the distributed processing frameworks Apache Hadoop/Spark/Flink and DataFlow and has GPU support.

Strong points

– High execution speed and model performance.
– Parallelization of tree construction using all of CPU cores during training.
– Distributed computing for training very large models using a cluster of machines.
– Out-of-core computing for very large datasets that do not fit into memory.
– Cache optimization of data structures and algorithms to make best use of hardware.

Weak points

- It is a boosting library that is designed for tabular data. Therefore it will not work for others tasks as NLP or computer vision.

The latest release of XGBoost is the version 0.80 (August 2018), which provides major upgrades on refactoring the design of XGBoost4J-Spark for JVM packages, improvements of GPU and Python support, and a number of new functionalities such as query ID column support in LibSVM data files or hinge loss for binary classification.

4.1.9 Interactive data analytic and visualization tools

Tools in this category display analytics results in an interactive way, so that they can ease the understanding of difficult concepts and support decision makers. There are many data visualization packages at various levels of abstraction in R or Python; e.g., `matplotlib`, `ggplot`, `seaborn`, `plotly`, `bokeh`.

In recent years, web-based notebooks/applications have gained in popularity. They are integrated with data analytic environments to create and share documents that contain data-driven live code, equations, visualizations and narrative text. The most well-known are:

- `Jupyter` notebook (Jupyter 2018) (formerly iPython notebook) is an open-source application supporting; e.g., creation and sharing documents (notebooks), code, source equations, visualization and text descriptions for data transformation, numerical simulations, statistical modeling, data visualization and ML. It has recently launched `JupyterLab` which aims to take this a step further.
- `Zeppelin` is an interactive notebook designed for the processing, analysis and visualization of large data sets (Zeppelin 2018). It provides native support for Apache Spark distributed computing. Zeppelin allows to extend their functionality through various interpreters; e.g., Spark, SparkSQL, Scala, Python, shell from Apache Spark analytics platform.

There are also open-source tools for data analytics, reporting and integration platforms such as:

- `Kibana` is the data visualisation front end for the Elastic Stack, complementing the rest of the stack that includes Beats, Logstash and Elasticsearch (Kibana 2018). With the version 5.x release of the Elastic Stack, Kibana now includes Timelion for interactive time series charts.
- `Grafana` is the DevOps tool for many real time monitoring dashboards of time series metrics (Grafana 2018). It offers visualisation and supports multiple backend data sources including InfluxDB, Graphite, Elasticsearch and many others which can be added via plugins.
- `Tableau` is a universal analytics tool, which can extract data from different small data sources like csv, excel, and SQL as well as from enterprise resources or connect Big Data frameworks and cloud based sources (Tableau 2018).

In short, there is a rich ecosystem of interactive tools, which are designed for different purposes.

4.1.10 Other data analytic frameworks and libraries

The number of frameworks and libraries providing or using ML/NN/DL techniques is high. A relevant subset of them are described below.

- `MatLab` is a multi-paradigm numerical computing environment. It uses a proprietary programming language developed by MathWorks (MatLab 2018). MatLab is quite popular with over 2 million users across industry and academia. On the other hand, MatLab is a proprietary product of MathWorks, so users are subject to vendor lock-in and future development will be tied to the MatLab language. The two most popular free alternatives to MatLab are GNU `Octave` (Octave 2018) and `SciLab` (SciLab 2018).
- `SAS` (Statistical Analysis System) began as a project to analyse agricultural data at North Carolina State University in 1966 (SAS 2018). Currently, it is a proprietary software package written in C for advanced data analytics and business intelligence with more than 200 components. Another similar proprietary software package is `SPSS` (Statistical Package for the Social Sciences) (SPSS 2018). It was developed in 1968 and was acquired by IBM in 2009. An open-source alternative of SPSS is GNU `PSPP` (PSPP 2018).
- `R` is a free software environment for statistical computing and graphics including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS (Rproject 2018). R is easy of use and extensible via packages. The Comprehensive R Archive Network offers more than 10 thousands packages, and the list is getting longer (R-CRAN 2018). It is important to notice that lots of frameworks have bindings for R.
- `Python` is a programming language created by Guido van Rossum and first released in 1991 (Python 2018). Python is successfully used in thousands of real-world business applications around the world e.g., Google and YouTube. The primarily rationale for adopting Python for ML is because it is a general purpose programming language for research, development and production, at small and large scales. Python features a dynamic type system and automatic memory management, with a large and comprehensive libraries for scientific computation and data analysis. As well as for R, lots of frameworks have bindings for Python.
- `NumPy` is the fundamental package for scientific computing with Python (NumPy 2018). Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. NumPy stack has similar users to MatLab, GNU Octave, and SciLab.
- `SciPy` is an open-source Python library used for scientific computing and technical computing (SciPy 2018). SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, Pandas, and SymPy.
- `Pandas` is a Python package providing fast, flexible, and expressive data structures designed to make it easier to work with relational or labelled data (Pandas 2018). Its two primary data structures, Series (one-dimensional) and DataFrame (two-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.
- `NLTK` (NLTK 2018) and `Gensim` (Rehurek 2018) are two leading toolkits for working with human language data. NLTK comes with a comprehensive list of text processing libraries for classification, tokenisation, stemming, tagging, parsing, and semantic reasoning. Gensim is an open-source vector space modeling and topic modeling designed for large text collections using data streaming and incremental algorithms. It is implemented in Python using NumPy, SciPy and Cython for performance.

The difference between Python and R is largely philosophical (Piatetsky 2017): Python is a general-purpose language designed by programmers for programmers; R was built for statistical analysis. The trends (such as Google trend, KDD trend) shows that R was slightly

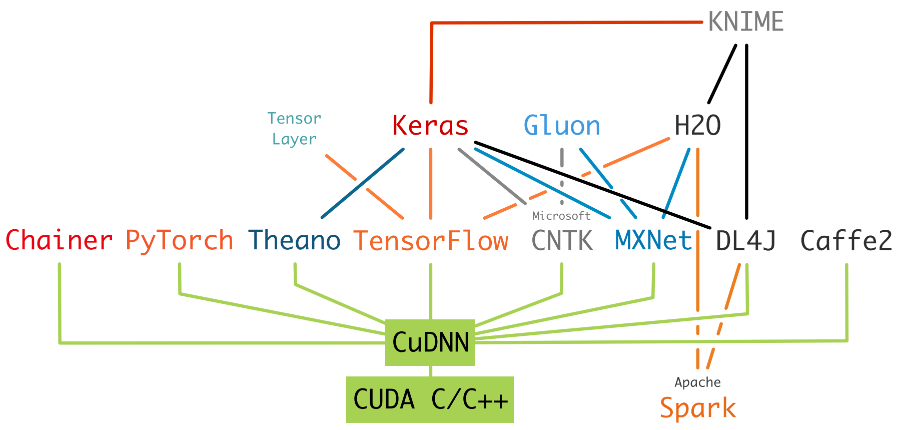

Fig. 3 The most popular Deep Learning frameworks and libraries layering in various abstraction implementation levels

ahead in 2014 and 2015 in data science and Machine Learning. *Since 2016, Python is clearly the most popular programming language in this area.*

4.2 Deep Learning frameworks and libraries

Many popular ML frameworks and libraries already offer the possibility to use GPU accelerators to speed up learning process with supported interfaces (DLwiki 2018; Felice 2017; Kalogeiton et al. 2016). Some of them also allow the use optimised libraries such as CUDA (cuDNN), and OpenCL to improve the performance even further. The main feature of many-core accelerators is that their massively parallel architecture allows them to speed up computations that involve matrix-based operations.

The software development in the ML/DL direction community is highly dynamic and has various layers of abstraction as depicted in Fig. 3. An overview of the most popular DL frameworks and libraries is presented in Table 4.

4.2.1 TensorFlow

TensorFlow is an open-source software library for numerical computation using data flow graphs (TensorFlow 2018). TensorFlow was created and is maintained by the Google Brain team within Google's Machine Intelligence research organization for ML and DL. It is currently released under the Apache 2.0 open-source license.

TensorFlow is designed for large-scale distributed training and inference. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The distributed TensorFlow architecture contains distributed master and worker services with kernel implementations. These include 200 standard operations, including mathematical, array manipulation, control flow, and state management operations written in C++. TensorFlow was designed for use both in research, development and production systems. It can run on single CPU systems, GPUs, mobile devices and large-scale distributed systems of hundreds of nodes.

In addition, *TensorFlow Lite* is the lightweight solution for mobile and embedded devices (TensorFlowLite 2018). It enables on-device ML inference with low latency and a small

Table 4 DL frameworks and libraries with GPU support

Tool	Licence	Written in	Computation graph	Interface	Popularity	Usage	Creator (notes)
TensorFlow (Numerical framework)	Open source, Apache 2.0	C++, Python	Static with small support for dynamic graph	Python, C++[a], Java[a], Go[a]	Very High Growing very fast	Academic Industrial	– **Google**
Keras (Library)	Open source, MIT	Python	Static	Python Wrapper for TensorFlow, CNTK, DL4J, MXNet, Theano	High Growing very fast	Academic Industrial	F. Chollet
CNTK (Framework)	Open source, Microsoft permissive license	C++	Static	Python, C++, BrainScript, ONNX	Medium Growing fast	Academic Industrial Limited mobile solution	– **Microsoft**
Caffe (Framework)	Open source, BSD 2-clause	C++	Static	C++, Python, MatLab	High Growing fast	Academic Industrial	Y. Jia **BAIR**
Caffe2 (Framework)	Open source, Apache 2.0	C++	Static	C++, Python, ONNX	Medium-low Growing fast	Academic Industrial Mobile solution	Y. Jia **Facebook**
Torch (Framework)	Open source, BSD	C++, Lua	Static	C, C++, LuaJIT, Lua, OpenCL	Medium–low Growing low	Academic Industrial	R. Collobert, K. Kavukcuoglu, C. Farabet

Table 4 continued

Tool	Licence	Written in	Computation graph	Interface	Popularity	Usage	Creator (notes)
PyTorch (Library)	Open source, BSD	Python, C	Dynamic	Python, ONNX	Medium Growing very fast	Academic Industrial	A. Paszke, S. Gross, S. Chintala, G. Chanan
MXNet (Framework)	Open source, Apache 2.0	C++	Dynamic dependency scheduler	C++, Python, Julia, MatLab, Go, R, Scala, Perl, ONNX	Medium Growing fast	Academic Industrial	– **Apache**
Chainer (Framework)	Open source, Owners permissive license	Python	Dynamic	Python	Low Growing low	Academic Industrial	– **Preferred Networks**
Theano (Numerical framework)	Open source, BSD	Python	Static	Python	Medium-low Growing low	Academic Industrial	Y. Bengio **University of Montreal**

[a]Not fully covered

binary size but has coverage for a limited set of operators. It also supports hardware acceleration with the Android Neural Networks API.

TensorFlow programming interfaces include APIs for Python and C++ and developments for Java, GO, R, and Haskell are on the way. TensorFlow is also supported in Google and Amazon cloud environments.

Strong points

- By far the most popular DL tool, open-source, fast evolving, supported by a strong industrial company (Google).
- Numerical library for dataflow programming that provides the basis for DL research and development.
- Efficiently works with mathematical expressions involving multi-dimensional arrays.
- GPU/CPU computing, efficient in multi-GPU settings, mobile computing, high scalability of computation across machines and huge data sets.

Weak points

- Still lower level API difficult to use directly for creating DL models.
- Every computational flow must be constructed as a static graph, although the TensorFlow Fold package (Google-AI-blog 2017) tries to alleviate this problem (Patel 2018).

The pre-release version of TensorFlow is 1.11.0-rc1 (September 2018) which fixes a performance issues from the previous version when training a Keras model in Eager mode. In the roadmap, TensorFlow 2.0 (announced in September 2018) is focused on ease of use (stronger integration with higher level APIs such as Keras, Eager and Estimators) and eager execution (distributed training on multi-GPU, multi-TPU, multi-machine as well as performance improvements), building out a set of reference models, etc. (TensorFlowCommunity 2018). Furthermore, Google TPU 3.0, which is $8\times$ more powerful than in 2017 with up to 100 petaFLOPS, is optimized for TensorFlow (GoogleTPU 2018).

4.2.2 Keras

Keras is Python wrapper library that provides bindings to other DL tools such as TensorFlow, CNTK, Theano, beta version with MXNet and announced Deeplearning4j (Keras 2018). It was developed with a focus on enabling fast experimentation and is released under the MIT license. Keras runs on Python 2.7 to 3.6 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. Keras is developed and maintained by Francois Chollet using four guiding principles:

1. *User friendliness and minimalism* Keras is an API designed with user experience in mind. Keras follows best practices for reducing cognitive load by offering consistent and simple APIs.
2. *Modularity* A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules to combine and to create new models.
3. *Easy extensibility* New modules are simple to add, and existing modules provide ample examples allowing to reduce expressiveness.
4. *Work with Python* Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

Strong points

- Open-source, fast evolving, with backend tools from strong industrial companies like Google and Microsoft.
- Popular API for DL with good documentation.
- Clean and convenient way to quickly define DL models on top of backends (e.g. TensorFlow, Theano, CNTK). Keras wraps backend libraries, abstracting their capabilities and hiding their complexity.

Weak points

- Modularity and simplicity comes at the price of being less flexible. Not optimal for researching new architectures.
- Multi-GPU still not 100% working in the terms of efficiency and easiness as pointed out by several benchmarks that used it with a TensorFlow backend (Zamecnik 2017; Vryniotis 2018; Kalogeiton et al. 2016).

The latest release is Keras 2.2.2 (July 2018).

4.2.3 Microsoft CNTK

Microsoft Cognitive Toolkit (CNTK) is a commercial grade distributed DL framework with large-scale datasets from Microsoft Research (CNTK 2018).

It implements efficient DNNs training for speech, image, handwriting and text data. Its network is specified as a symbolic graph of vector operations, such as matrix add/multiply or convolution with building blocks (operations). CNTK supports FFNN, CNN, RNN architectures and implements stochastic gradient descent (SGD) learning with automatic differentiation and parallelization across multiple GPUs and servers.

CNTK is running on both 64-bit Linux and Windows operating systems using Python, C#, C++ and BrainScript API.

Strong points

- Open-source, fast evolving, supported by a strong industrial company (Microsoft).
- Supports the Open Neural Network Exchange (ONNX) format, which allows to easily transform models between CNTK, Caffe2, PyTorch, MXNet and other DL tools. ONNX is co-developed by Microsoft and Facebook.
- Higher performance (speed) in comparison with Theano and TensorFlow when used as Keras backend on multiple machines for RNN/LSTM in several benchmarks (Lee 2017; Woolf 2017; Bhatia 2017).

Weak points

- Limited capability on mobile devices.

The latest release is CNTK version 2.6 (September 2018).

4.2.4 Caffe

Caffe is a DL framework made with expression, speed, and modularity in mind. It is developed by Yangqing Jia at BAIR (Berkeley Artificial Intelliegence Research) and by community contributors (BAIR 2018).

DNNs are defined in Caffe layer-by-layer. Layer is the essence of a model and the fundamental unit of computation. Data enters Caffe through data layers. Accepted data sources are efficient databases (LevelDB or LMDB), Hierarchical Data Format (HDF5) or common image formats (e.g. GIF, TIFF, JPEG, PNG, PDF). Common and normalization layers provide various data vector processing and normalisation operations. New layers must be written in C++ CUDA, although custom layers are also supported in Python (but are less efficient).

Strong points

- Suitable for image processing with CNNs.
- Pretrained networks are available in the Caffe Model Zoo for finetuning.
- Easy to code (API/CLI) with Python and MatLab interface.

Weak points

- Development is not as active as previously.
- Static model graph definition does not fit many RNN applications which need variable sized inputs.
- Model definition in Caffe prototxt files is overly cumbersome for very deep and modular DNN models, such as GoogleLeNet or ResNet in comparison with other frameworks.
- Custom layers must be written in C++.

The latest available version of Caffe is 1.0 (April 2017), which is a stable, reference release of the framework and a shift into maintenance mode with the next generation successor Caffe2. There are currently several custom distributions available as well; i.e., Intel Caffe (multi-node and selected Intel Xeon processor optimized version), OpenCL Caffe (yet experimental version with OpenCL backend and additional layers for fast image segmentation), Windows Caffe (experimental version for Windows and Visual Studio).

4.2.5 Caffe2

Caffe2 is a lightweight, modular, and scalable DL framework developed by Yangqing Jia and his team at Facebook (Caffe2 2018).

Although it aims to provide an easy and straightforward way to experiment with DL and leverage community contributions of new models and algorithms, Caffe2 is used at production level at Facebook while development is done in PyTorch. Caffe2 differs from Caffe in several improvement directions, namely by adding mobile deployment and new hardware support (in addition to CPU and CUDA). It is headed towards industrial-strength applications with a heavy focus on mobile. The basic unit of computation in Caffe2 is the operator, which is a more flexible version of Caffe's layer. There are more than 400 different operators available in Caffe2 and more are expected to be implemented by the community.

Caffe2 provides command line Python scripts capable of translating existing Caffe models into the Caffe2. However, the conversion process needs to perform a manual verification of the accuracy and loss rates. It is possible to convert Torch models to Caffe2 models via Caffe.

Strong points

- Cross-platform, focused also on mobile platform, edge device inference deployment framework of choice for Facebook.
- Amazon, Intel, Qualcomm, NVIDIA claim to support Caffe2 due to its robust scalable character in production.

– Supports the Open Neural Network Exchange (ONNX) format, which allows to easily transform models between CNTK, Caffe2, PyTorch, MXNet and other DL tools.

Weak points

– Harder for DL beginners in comparison with PyTorch (Caffe2PyTorch 2018).
– Without dynamic graph computation.

The current pre-released version is Caffe2 v0.8.1 (August 2018) and the installation is available for Mac OS X, Ubuntu, CentOS, Windows, iOS, Android, Raspbian, and Tegra. At the moment, the installation supports only Anaconda packages. For other python packages, Caffe2 can be built from source. Caffe2 will be merged with PyTorch in order to combine the flexible user experience of the PyTorch frontend with the scaling, deployment and embedding capabilities of the Caffe2 backend.

4.2.6 Torch

Torch is a scientific computing framework with wide support for ML algorithms based on the Lua programming language (Torch 2018). It has been under active development since 2002 (Collobert et al. 2002). Torch is supported and used by Facebook, Google, DeepMind, Twitter, and many other organizations and it is freely available under a BSD license. It uses an object-oriented paradigm and is implemented in C++. Nowadays, its API is also written in Lua, which is used as a wrapper for optimized C/C++ and CUDA code.

Its core is made up by the Tensor library available both with CPU and GPU backends. The Tensor library provides a lot of classic operations (including linear algebra operations), efficiently implemented in C, leveraging SSE instructions on Intel's platforms and optionally binding linear algebra operations to existing efficient BLAS/Lapack implementations (like Intel MKL) (Collobert et al. 2011). The framework supports parallelism on multi-core CPUs via OpenMP, and on GPUs via CUDA. It is mainly used for large-scale learning (speech, image, and video applications), supervised learning, unsupervised learning, reinforcement learning, NNs, optimization, graphical models, image processing.

Strong points

– Flexibility, readability, mid-level code as well as high level (Lua), easy code reuse.
– Modularity and speed.
– Very convenient for research.

Weak points

– Still smaller proportion of projects than Caffe.
– LuaJIT is not mainstream and does cause integration issues and Lua is not popular although it is easy to learn.
– No longer under development.

Torch is no longer in active development, the latest version is Torch7.

4.2.7 PyTorch

PyTorch is a Python library for GPU-accelerated DL (PyTorch 2018). The library is a Python interface of the same optimized C libraries that Torch uses. It has been developed by Facebook's AI research group since 2016.

PyTorch is written in Python, C and CUDA. The library integrates acceleration libraries such as Intel MKL and NVIDIA (cuDNN, NCCL). At the core, it uses CPU and GPU Tensor and NN backends (TH, THC, THNN, THCUNN) written as independent libraries on a C99 API.

PyTorch supports tensor computation with strong GPU acceleration, and DNNs built on a tape-based autograd system. It has become popular by allowing complex architectures to be built easily. Typically, changing the way a network behaves means to start from scratch. PyTorch uses a technique called *reverse-mode auto-differentiation*, which allows to change the way a network behaves with small effort (i.e. *dynamic computational graph* or DCG). It is mostly inspired by autograd (autograd 2018), and Chainer (Chainer 2018).

The library is used by both the scientific and industrial communities. An engineering team at Uber has built *Pyro*, a universal probabilistic programming language that uses PyTorch as backend. The DL training site `fast.ai` announced that their courses will be based on PyTorch rather than Keras-TensorFlow (Patel 2017). The library is freely available under a BSD license and is supported by Facebook, Twitter, NVIDIA, and many other organizations.

Strong points

– Dynamic computational graph (reverse-mode auto-differentiation).
– Supports automatic differentiation for NumPy and SciPy.
– Elegant and flexible Python programming for development (Caffe2PyTorch 2018).
– Supports the Open Neural Network Exchange (ONNX) format, which allows to easily transform models between CNTK, Caffe2, PyTorch, MXNet and other DL tools.

Weak points

– Still without mobile solution, although this is going to be taken care of in the 1.0 release (PyTorchTeam 2018) with a closer integration with Caffe2 which will enable to create models in PyTorch and deploy them in production with Caffe2 thanks to a JIT compiler.

As of September 2018, PyTorch is about to release its 1.0 version (currently in alpha). It will include among other things tighter integration with Caffe2 and ONNX to enable production readiness of PyTorch models.

4.2.8 MXNet

Apache MXNet is a DL framework designed for both efficiency and flexibility (MXNet 2018). It is developed by Pedro Domingos and a team of researchers at the University of Washington, it is also a part of the DMLC (DMLC 2018).

It allows mixing symbolic and imperative programming to maximize efficiency and productivity. At its core, MXNet contains a *dynamic dependency scheduler* that automatically parallelizes both symbolic and imperative operations *on-the-fly*. A graph optimization layer on top of that makes symbolic execution fast and memory efficient. MXNet is portable and lightweight, scaling effectively to multiple GPUs and multiple machines. It also supports an efficient deployment of trained models in low-end devices for inference, such as mobile devices (using Amalgamation), IoT devices (using AWS Greengrass), Serverless (using AWS Lambda) or containers.

MXNet is licensed under an Apache-2.0 license and has a broad API language support for R, Python, Julia and other languages (Chen et al. 2015). MXNet is supported by major public cloud providers.

Strong points

- Dynamic dependency scheduler (auto parallelism).
- Very good computational scalability with multiple GPUs and CPUs, which makes it very useful for the enterprises.
- Supports a flexible programming model and multiple languages i.e. C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl, and Wolfram.
- Supports the Open Neural Network Exchange (ONNX) format, which allows to easily transform models between CNTK, Caffe2, PyTorch, MXNet and other DL tools.

Weak points

- In some cases, APIs are not very user-friendly (althouh Keras and Gluon can be used).

The last version of Apache MXNet (incubating) is 1.3.0 (September 2018). It includes improved MXNet Scala API with new examples, MXNet to ONNX exporter APIs, MKL-DNN, new runtime integration of TensorRT into MXNet, sparse tensor support for Gluon, experimental topology-aware AllReduce and Clojure package. The MXNet roadmap is to improve performance and scalability for distributed training for Gluon CV, NLP toolkit, Android SDK as well as to enhance supports for low-bit precision inference, IoT device inferencing and MKL-DNN RNN API.

4.2.9 Chainer

Chainer is a Python-based, standalone open-source framework for DL models (Chainer 2018). Chainer core team of developers work at Preferred Networks, Inc., a ML startup with engineers mainly from the University of Tokyo.

It provides a full range of DL models, including CNN, RNN, reinforcement learning (RL), and variational autoencoders. The Chainer vision is going beyond invariance (Tokui et al. 2015).

Chainer provides automatic differentiation APIs based on the *Define-by-Run* approach; i.e., *dynamic computational graphs* (DCG) as well as object-oriented high-level APIs to build and train NNs. Chainer constructs NN dynamically (computational graph is constructed *on-the-fly*), while other frameworks (such as TensorFlow of Caffe) build their graph according to the *Define-and-Run* scheme (graph is constructed at the beginning and remains fixed).

Chainer supports CUDA/cuDNN using CuPy, for high performance training and inference, and the Intel Math Kernel Library (Intel MKL) for Deep Neural Networks (MKL-DNN), which accelerates DL frameworks on Intel based architectures. It also contains libraries for industrial applications; e.g., ChainerCV (for computer vision), ChainerRL (for deep reinforcement learning) and ChainerMN (for scalable multi-node distributed DL) (ChainerMN 2018).

In the benchmark performed by Akiba (2017), ChainerMN showed the best performance in a multi-node setting (4 GPUs per node, up to 128 GPUs) where it outperformed MXNet, CNTK and TensorFlow for ImageNet classification using ResNet-50.

Strong points

- Dynamic computational graph based on the *Define-by-Run* principle.
- Provides libraries for industrial applications.
- Strong investors such as Toyota, FANUC, NTT.

Weak points

- No support for higher order gradients.
- DCG is generated every time also for fixed networks.

In the latest release v5.0.0rc1 (September 2018), ChainerMN is already integrated into the Chainer package. The version included a number of new features; e.g., experimental static subgraph optimization, support for Intel architecture and NVIDIA Dali to construct data preprocessing pipeline.

4.2.10 Theano

Theano is a pioneering DL tool supporting GPU computation whose development started in 2007. It is an open-source project released under the BSD license (Theano 2018). It is actively maintained (although no longer developed) by the LISA group [now MILA Montreal Institute for Learning Algorithms (MILA 2018)] at the University of Montreal.

At its heart, Theano is a compiler for mathematical expressions in Python to transform structures into very efficient code using NumPy and efficient native libraries like BLAS and native code to run as fast as possible on CPUs or GPUs. Theano supports extensions for multi-GPU data parallelism and has a distributed framework for training models.

Strong points

- Open-source, cross-platform project.
- Powerful numerical library that provides the basis for DL research and development.
- Symbolic API supports looping control, which makes implementing RNNs efficient.

Weak points

- Lower level API, difficult to use directly for creating DL models, although wrappers (like Lasagne or Keras) exist.
- Lack for mobile platform and other programming API's.
- No longer under active development.

The active development of Theano ended at the 1.0.0 final release version on November 2017 as announced in Bengio (2017). The Theano maintenance continue as the current version is 1.0.3 (September 2018).

4.2.11 Performance-wise preliminary

Under the assumption of the same datasets, methods and hardware, there are two concerns about performance of DL frameworks and libraries: model performance (mean of accuracy of the model) and runtime performance (mean of speed of training/inference). While the model performance is always in the first place of interests, the DL community has put considerable effort into benchmarking and comparing the runtime performance of different libraries and frameworks. In recent years, a non-exhaustive list of benchmarks has appeared:

- Comparing the most well-known DL frameworks (Bahrampour et al. 2015; Shi et al. 2016; Akiba 2017; Karmanov et al. 2018; Liu et al. 2018) (e.g. TensorFlow, CNTK, PyTorch, Caffe2, Chainer, Theano, MXNet), with and without wrapper libraries like Keras or Gluon.

- Testing Keras back-ends (Woolf 2017; Lee 2017; Bhatia 2017) (e.g. among TensorFlow, Theano, CNTK and MXNet). The comparison with the DL4J back-end (which has a Java API integrated with Hadoop/Spark, and therefore targets a different community) is still missing.
- Comparing Keras against PyTorch (Migdal and Jakubanis 2018).
- Benchmarking one method, like CNNs (Chintala 2017) or LSTMs (Braun 2018), with various DL frameworks.
- Studying a number of distributed DL frameworks (like Caffe, Chainer, CNTK, MXNet and TensorFlow) on CNNs for a number of parameters like setup, code conversion for multiple nodes, functionality, popularity according to GitHub, performance, memory utilization and scalability (Liu et al. 2018; Akiba 2017).

The most frequently used datasets for benchmarking are the IMDb review dataset (IMDb 2018) for NLP and sentiment analysis, the MNIST (LeCun 1998), CIFAR-10 (Krizhevsky 2009) and ImageNet (Russakovsky et al. 2015) datasets for image classification. However, the number of publicly accessible datasets is increasing every year. Modern datasets with highly complex scenes and situations, such as Microsoft COCO for image detection, segmentation and keypoints (Lin et al. 2014) or Cityscape for urban scene understanding and autonomous driving image recognition (Cordts et al. 2016), are currently used in scientific comparisons.

The most frequently used DL architectures for benchmarking are CNN (e.g. ResNet, AlexNet, and GoogleNet), multilayer perceptrons, fully connected NNs, RNNs (LSTM, GRU) and stacked autoencoders.

The most popular GPU models for benchmarking include NVIDIA Tesla K80, NVIDIA GTX 1080, NVIDIA Tesla P100 and NVIDIA Titan X, usually available through Cloud services.

Results of these benchmarks show similar accuracy for almost all the frameworks, while the runtime performance can vary sometimes. For example, Chainer outperforms MXNet, CNTK and TensorFlow in benchmark with ResNet-50 (Akiba 2017). Regarding RNNs, CNTK and PyTorch often hit the top scores alternately, but other tools like MXNet and Chainer perform well as well. As always, it is often difficult to assess how much of this difference is due to the frameworks themselves and how much is due to the correct implementation of the model in particular framework. In any case, *in principle*, there should not be a big difference in terms of runtime performance as the most frameworks use the same underlying cuDNN primitives.

It is important to bear in mind that *there is no clear winner for all use cases* due to the sensitiveness to different choices and problem settings (Liu et al. 2018). Although some benchmarks feature an impressive list of testing experiments with various hardware/datasets/methods, they are not intended to give the overall performance of frameworks. The results just show comparisons over specific cases across different frameworks and hardware and are subject to change over time with new hardware and libraries updates.

4.2.12 Deep Learning wrapper libraries

As mentioned above, Keras is a wrapper library for DL libraries intended to hide low level implementations. Other wrapper libraries are:

- TensorFlow has a lot of wrappers. External wrapper packages are `Keras`, `TensorLayer` (TensorLayer 2018), and `TFLearn` (TFLean 2018). Wrappers from Google Deepmind are `Sonnet` (Sonnet 2018) and `PrettyTensor` (Tensor 2018). Wrappers within native

TensorFlow are `TF-Slim` (TFSlim 2018), `tf.keras`, `tf.contrib.learn`, and `tf.layers` (TensorFlow 2018).

– `Gluon` is a wrapper for MXNet (Gluon 2018). Gluon's API specification is an effort to improve speed, flexibility, and accessibility of DL technology for all developers, regardless of their DL framework choice. Gluon is a product by Amazon Web Services and Microsoft AI. It is released under Apache 2.0 licence.
– `NVIDIA Digits` (Digits 2018) is a web application for training DNNs for image classification, segmentation and object detection tasks using DL backends such as Caffe, Torch and TensorFlow with a wide variety of image formats and sources with Digits plugins. It simplifies common DL tasks such as managing data, designing and training NNs on multi-GPU systems, monitoring performance in real time with advanced visualisations, and selecting the best performing model for deployment based on the results browser. The tool is mainly interactive (GUI) with a number of pre-trained models; e.g., AlexNet, GoogLeNet, LeNet and UNet from the Digits Model Store. Digits is released under BSD 3-clause license.
– `Lasagne` is a lightweight library to build and train NNs in Theano with six principles: Simplicity, Transparency, Modularity, Pragmatism, Restraint and Focus (Lasagne 2018). Other wrappers for Theano are `Blocks` and `Pylearn2`.

The development of DL frameworks and libraries is highly dynamic and therefore makes it difficult to forecast who will lead this fast changing ecosystem but we can see two main trends emerging in the use of DL frameworks:

1. Using Keras for fast prototyping and TensorFlow for production. This trend is backed by Google.
2. Using PyTorch for prototyping and Caffe2 for production. This trend is backed by Facebook.

The extensive number of Deep Learning frameworks makes it challenging to develop tools in one framework and use them in other frameworks (framework interoperability). The Open Neural Network Exchange [ONNX] tries to address this problem by introducing an open ecosystem for interchangeable AI models. ONNX is being co-developed by Microsoft, Amazon and Facebook as an open-source project and it will initially support Caffe2, PyTorch, MXNet and CNTK.

4.3 Machine Learning and Deep Learning frameworks and libraries with MapReduce

Recently, newly distributed frameworks have emerged to address the scalability of algorithms for Big Data analysis using the MapReduce programming model, being Apache Hadoop and Apache Spark the two most popular implementations. The main advantages of these distributed systems is their elasticity, reliability, and transparent scalability in a user-friendly way. They are intended to provide users with easy and automatic fault-tolerant workload distribution without the inconveniences of taking into account the specific details of the underlying hardware architecture of a cluster. These popular distributed computing frameworks are not mutually exclusive technologies with GPUs, although they aim at different scaling purposes (Cano 2018). These technologies can complement each other and target complementary computing scopes such as ML and DL (Skymind 2017). However, there is still a lot of limitations and challenges.

Table 5 ML/DL frameworks and libraries with MapReduce

Tool	Licence	Written in	Backends	Interface	Popularity	Algorithm coverage	Usage	Creator
DL4J (DL library for Java)	Open source, Apache 2.0	Java, Scala CUDA cuDNN support via JNI	Integration with Spark	Java, Scala, Clojure, Python	Medium	Medium (DL)	Industrial	**Skymind**
Spark MLlib[a], Spark ML[a] (ML/NN library)	Open source, Apache 2.0	Scala	Integration with Python, R	Java, Scala, Python, R	High for Spark, low for ML	Medium (ML)	Industrial	**Apache**
H2O (ML/DL framework)	Open source, Apache 2.0	Java	TensorFlow, MXNet, Caffe	REST API, JSON+HTTP, Java, Scala, Python, R	Medium	Medium (ML/DL)	Industrial	**H2O**
KNIME (Analytic platform)	Open source, GNU GPLv3	Java with CUDA support	R, Python, Weka, Keras, H2O, DL4J	GUI wrapper	Low	as backends	Academic Industrial	**KNIME.ai**

[a] Apache Spark MLLib and ML are parts of the Apache Spark repositories so they do not have separate insights (Spark 2018b)

4.3.1 Deeplearning4j

Deeplearning4j or DL4J differs from other ML/DL frameworks in its API languages, intent and integrations. It is a modern open-source, distributed, DL library implemented in Java (JVM) aimed to the industrial Java development ecosystem and Big Data processing. DL4J framework comes with built-in GPU support, which is an important feature for the training process and supports Hadoop and Spark (DL4J 2018; Skymind 2017). The library consists of several sub-projects such as raw data transformation into feature vectors (DataVec), tools for NN configuration (DeepLearning4j), 3rd party model import (Python and Keras models), native libraries support for quick matrix data processing on CPU and GPU (ND4J), Scala wrapper running on multi-GPU with Spark (ScalNet), library of reinforcement learning algorithms (RL4J), tools for searching the hyperparameter space to find the best NN configuration, and working examples (DL4J-Examples). Deeplearning4j has Java, Scala and also Python APIs.

It supports various types and formats of input data easily extendable by other specialized types and formats. The DataVec toolkit accepts raw data such as images, video, audio, text or time series on input and enables its ingestion, normalization and transformation into feature vectors. It can also load data into Spark RDDs. DataVec contains record readers for various common formats. DL4J includes some of the core NLP tools such as SentenceIterator (for feeding text piece by piece into a natural language processor), Tokenizer (for segmenting the text at the level of single words or n-grams), Vocab (cache for storing metadata). Specialized formats can be introduced by implementing custom input formats similarly as in Hadoop via InputFormat.

Strong points

– Great advantage of DL4j is that it uses the whole potential of the Java ecosystem to perform efficient DL (Varangaonkar 2017). It can be implemented on top of the popular Big Data tools such as Apache Hadoop, Spark, Kafka with an arbitrary number of GPUs or CPUs. DL4J is the choice for many commercial industry-focused distributed DL platforms, where the Java ecosystem is predominant.
– Provides a native model zoo containing DL models with pretrained weights for different datasets.

Weak points

– Java/Scala are not as popular in the DL/ML research community as Python.
– Currently, it gains less overall interest than H2O in Big Data and Spark community.

The latest version of DL4J is 1.0.0-beta2 (September 2018), adding support for CUDA 9.2 and dropping it for CUDA 9.1. Its new functionalities include the ability to import Keras applications, support for all Keras optimizers and advanced activation functions, and an automatic differentiation package, called SameDiff, analogous to how TensorFlow and PyTorch to calculate gradients for NNs training. This version replaced OpenBLAS with Intel MKL-DNN.

4.3.2 Apache Spark MLlib and Spark ML

At the beginning, Apache introduced *Mahout* built on the top of MapReduce. Mahout was mature and came with many ML algorithms. However, in general ML algorithms need many iterations making the Mahout run very slow. Therefore, Apache introduced *Spark MLlib* and *Spark ML* built on top of Spark ecosystem (Spark 2018b) thus being much faster than Mahout.

Spark MLlib contains old RDD-based API (Resilient Distributed Dataset). RDD is the Spark's basic abstraction of data representing an immutable, partitioned collection of elements that can be operated on in parallel with a low-level API that offers transformations and actions. Spark ML contains new API build around DataFrame-based API and ML pipelines and it is currently the primary ML API for Spark. A DataFrame is a Dataset organised into named columns and it is conceptually equivalent to a table in a relational database. Transformations and actions over a DataFrame can be specified as SQL queries, which is convenient for developers with SQL background. Moreover, Spark SQL provides to Spark more information about the structure of both the data and the computation being performed than Spark RDD API. Spark ML brings the concept of ML pipelines, which help users to create and tune practical ML pipelines; it standardises APIs for ML algorithms so multiple ML algorithms can be combined into a single pipeline, or workflow. The MLlib is now in maintenance mode and the primary Machine Learning API for Spark is the ML package.

Spark MLlib/ ML contains Machine Learning algorithms such as classification, regression, clustering or collaborative filtering; featurization tools for feature extraction, transformation, dimensionality reduction and selection; pipeline tools for constructing, evaluating and tuning ML pipelines; and persistence utilities for saving and loading algorithms, models and pipelines. It also contains tools for linear algebra, statistics and data handling. With the exception of the distributed data parallel model, MLlib can be easily used together with stream data as well. For this purpose, MLlib offers few basic ML algorithms for stream data such as streaming linear regression or streaming k-means. For a larger class of ML algorithms, one have to let model learn offline and then apply the model on streaming data online.

It is important to notice that the implementation of a seemingly simple algorithm (e.g. distributed multi-label kNN) for large-scale data mining is not trivial (Ramirez-Gallego et al. 2017; Gonzalez-Lopez et al. 2018). It requires deep understanding about underlying scalable and distributed environment (e.g. Apache Spark), its data and processing management as well as programming skills. Therefore, ML algorithms for large-scale data mining are different in complexity and implementation from general purpose ones.

Strong points

- ML tools for large-scale data, which are already integrated in Apache Spark ecosystem, convenient to use in development and production.
- Optimized selected algorithm with optimized implementations for Hadoop included pre-processing methods.
- Pipeline (workflow) building for Big Data processing included a set of feature engineering functions for data analytics (classification, regression, clustering, collaborative filtering and featurization) also with stream data.
- Scalability with SQL support and very fast because of the in-memory processing.

Weak points

- Mainly focused to work on tabular data.
- High memory consumption because of the in-memory processing.
- Spark MLlib/ML are quite young ML libraries in evolving state.

As of Spark 2.0, the RDD-based APIs in the spark.mllib package have entered maintenance mode. The primary Machine Learning API for Spark is now the DataFrame-based API in the spark.ml package. MLlib is currently adding features to the DataFrames-based API to reach feature parity with the RDD-based API. After reaching the parity, the RDD-based API will be deprecated and removed with expected announcement in Spark 3.0 (Spark 2018a).

4.3.3 H2O, Sparkling Water and Deep Water

H2O, Sparkling Water and Deep Water are developed by H2O.ai (formerly 0xdata) (H2O 2018; H2O.ai 2017). They are Hadoop compatible frameworks for ML and DL over Big Data as well as for Big Data predictive analytics.

To access and reference data, models and objects across all nodes and machines, H2O uses distributed key/value store. H2O's algorithms are implemented on top of distributed MapReduce framework and utilize the Java Fork/Join framework for multi-threading. H2O can interact in a stand-alone fashion with HDFS stores, on top of YARN, in MapReduce, or directly in an Amazon EC2 (Elastic Compute Cloud) instance. Hadoop mavens can use Java to interact with H2O, but the framework also provides REST API via JSON over HTTP and bindings for Python (H2O-Python), R (H2O-R), and Scala, providing cross-interaction with all the libraries available on those platforms as well. H2O also provides stacking and boosting methods for combining multiple learning algorithms in order to obtain better predictive performance.

Except the REST API and the bindings for popular programming languages, H2O is accessible through CLI as well giving possibilities to set several options to control cluster deployment such as how many nodes to launch, how much memory to allocate for each node, assign names to the nodes in the cloud, and more. It offers a web-based interactive environment called *Flow* (which is similar to Jupyter). Data sources for the framework are natively local FS, Remote File, HDFS, S3, JDBC, others through generic HDFS API. H2O ML algorithms are optimised to run over Big Data and cover the need of the target companies; i.e., banks and insurance sectors. H2O.ai has a strong partnership with companies such as Wells Fargo, Citigroup, Capital One, PayPal, Discover, Dun & Bradstreet and Equifax and its products are used by companies like PayPal, PwC and Brazilian banks (BusinessWire 2017).

DL in H2O it is based on FFNNs trained with stochastic gradient descent (SGD) using back-propagation. The global model is periodically built from local models via model averaging. Local models are build on each node with multi-threading using global model parameters and local data.

Sparkling Water contains the same features and functionality as H2O but provides a way to use H2O with Spark. It is ideal for managing large clusters for data processing, especially when it comes to transfer data from Spark to H2O (or vice versa).

Deep Water is H2O DL with native implementation of DL models for GPU-optimised backends such as TensorFlow, MXNet, and Caffe. These backends are accessible from Deep Water through connectors.

Strong points

– Industrial use with significant growth and high popularity among financial, insurance and healthcare companies.
– optimization algorithms for Big Data processing and analytics with infrastructure supports.

– H2O provides a generic set of ML algorithms that leverages Hadoop/Spark engines for large-scale dataset processing. It aims to make ML/DM process more automatic through GUI.

Weak points

– UI flow, the web-based UI for H2O does not support direct interaction with Spark.
– H2O is more general purpose and aims at a different problem in comparison with (specific) DL libraries e.g., TensorFlow or DL4j.

The latest stable version of H2O is 3.20.0.8 (September 2018).

4.3.4 Other frameworks and libraries with MapReduce

From the rest of the frameworks and libraries that work with MapReduce, we mention the most important ones:

– `FlinkML` is a part of Apache Flink, which is an open-source framework for distributed stream and batch data processing (Flink 2018). FlinkML aims to provide a set of scalable ML algorithms and API adopted to Flink distributed framework. It contains algorithms for supervised learning, unsupervised learning, data preprocessing, recommendation and other utilities. Flink is focused on working with lots of data with very low data latency and high fault tolerance on distributed systems. Its core feature is the ability to process data streams in real time. The main difference between Spark and Flink is in the way how each of the frameworks deals with streams of data. Flink is a native streaming processing framework that can work on batch data. Spark was originally designed to work with static data through its RDDs, it uses micro-batching to deal with streams.
– `Oryx 2` from Cloudera also has a ML layer. Oryx 2 is a realization of the Lambda architecture built on top of Apache Spark and Apache Kafka for real-time large-scale ML (Oryx2 2018). It is designed for building applications and includes packaged, end-to-end applications for collaborative filtering, classification, regression and clustering. Oryx 2 comprises the following three tiers: (1) general Lambda architecture tier for batch, speed and serving layers, which are not specific to ML; (2) ML abstraction to hyperparameter selection; (3) end-to-end implementation of the same standard ML algorithms as an application; e.g., k-means, random decision forests, alternating least squares.
– `KNIME` (Konstanz Information Miner) is the data analytic, reporting and integration platform of the Knime AG, Switzerland (KNIME 2018). It integrates various components for ML and DM within its modular data pipelining concept (GUI) allowing assembly of nodes for data preprocessing, modelling and data analysis and visualisation without, or with only minimal, programming. The platform is released under open-source GNU GPLv3 license and has more than 1500 modules, a comprehensive range of integrated tools, and the widest choice of advanced algorithms available. KNIME is implemented in Java but allows wrappers to call other code in addition to nodes that allow running Java, Python, Perl and other programming languages. It is integrated with Weka, R, Python, Keras, H2O and DL4J. It has considerable community support; i.e., it is used by over 3000 organizations in more than 60 countries.The latest stable version available for Window, Linux, and Mac OS X is KNIME 3.6.1 (July 2018) with support for H2O Sparkling Water.

5 Conclusions

Machine Learning and Deep Learning are research areas of computer science with constant developments due to the advances in data analysis research in the Big Data era. This work provides the comprehensive survey with detailed comparisons of popular frameworks and libraries that exploit large-scale datasets. This work could be summarized as follows:

1. Most of the Deep Learning frameworks are developed by the world's largest software companies such as Google, Facebook, and Microsoft. These companies possess huge amounts of data, high performance infrastructures, human intelligence and investment resources. Such tools include TensorFlow, Microsoft CNTK, Caffe, Caffe2, Torch, PyTorch, and MXNet. Apart from them, other Deep Learning frameworks and libraries such as Chainer, Theano, Deeplearning4J, and H2O from other companies and research institutions, are also interesting and suitable for industrial use.
2. There are many high level Deep Learning wrapper libraries built on top of the above-mentioned Deep Learning frameworks and libraries. Such wrappers include Keras, TensorLayer and Gluon.
3. Big Data ecosystems such as Apache Spark, Apache Flink and Cloudera Oryx 2 contain build-in Machine Learning libraries for large-scale data mining mainly for tabular data. These Machine Learning libraries are currently in an evolving state but the power of the whole ecosystem is significant.
4. Vertical scalability for large-scale DL is still limited due to the GPU memory capacity and horizontal scalability is still limited due to the network communication latency between nodes.
5. Every tool, including traditional general purpose Machine Learning tools, provides a way to process large-scale data.
6. As of 2018, Python is the most popular programming language for data mining, Machine Learning and Deep Learning applications. It is used as a general purpose language for research, development and production, at small and large scales. The majority of tools are either Python tools or support Python interfaces.
7. The trend shows a high number of interactive data analytics and data visualisation tools supporting decision makers.

The impact of new computing resources and techniques combined with an increasing avalanche of large datasets is transforming many research areas. This evolution has many different faces, components and contexts. Since the technology is more and more present, the domain knowledge is not sufficient to tackle complex problems. This brings a great challenge for data mining projects in deciding which tools to select among the myriad of frameworks, libraries, tools and approaches from divergent Machine Learning and Deep Learning user communities in different applicable areas.

Acknowledgements Álvaro López García, Ignacio Heredia, Giang Nguyen, Viet Tran, Stefan Dlugolinsky, Martin Bobák, and Ladislav Hluchý are supported by the project DEEP-HybridDataCloud "Designing and Enabling E-infrastructures for intensive Processing in a Hybrid DataCloud" that has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 777435. Giang Nguyen, Viet Tran, Stefan Dlugolinsky, Martin Bobák, and Ladislav Hluchý are also supported by the Project VEGA 2/0167/16 "Methods and algorithms for the semantic processing of Big Data in distributed computing environment". The authors would like to thanks to all colleagues, especially for Ján Astaloš for knowledge sharing and teamwork.

References

Abdiansah A, Wardoyo R (2015) Time complexity analysis of support vector machines (SVM) in LibSVM. Int J Comput Appl 128(3):28–34

Akiba T (2017) Performance of distributed deep learning using ChainerMN. https://chainer.org/general/2017/02/08/Performance-of-Distributed-Deep-Learning-Using-ChainerMN.html. Accessed 4 Oct 2018

Alcala-Fdez J, Fernandez A, Luengo J, Derrac J, Garcia S, Sanchez L, Herrera F (2011) Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult Valued Logic Soft Comput 17:255–287

Alemany S, Beltran J, Perez A, Ganzfried S (2018) Predicting hurricane trajectories using a recurrent neural network. arXiv preprint arXiv:1802.02548

AMD (2018) Accelerators for high performance compute. https://www.amd.com/en/products/servers-hpc-accelarators. Accessed 21 Sept 2018

Anaconda (2018) Anaconda—the most popular python data science platform. https://www.anaconda.com/what-is-anaconda/. Accessed 20 Oct 2018

AnacondaCloudera (2016) Anaconda for Cloudera—data science with python made easy for big data. http://know.continuum.io/anaconda-for-cloudera.html. Accessed 20 Oct 2018

Andres R et al (2018) Lower numerical precision deep learning inference and training. Intel. https://software.intel.com/en-us/articles/lower-numerical-precision-deep-learning-inference-and-training. Accessed 18 Sept 2018

autograd (2018) autograd—automatic differentiation—efficiently computes derivatives of numpy code. https://github.com/HIPS/autograd. Accessed 26 Feb 2018

Bachniak D, Rauch L, Krol D, Liput J, Slota R, Kitowski J, Pietrzyk M (2017) Sensitivity analysis on HPC systems with Scalarm platform. Concurr Comput 29(9):172–181

Bahrampour S, Ramakrishnan N, Schott L, Shah M (2015) Comparative study of deep learning software frameworks. arXiv preprint arXiv:1511.06435

BAIR (2018) Caffe—deep learning framework by Berkeley Artificial Intelligence Research (BAIR). http://caffe.berkeleyvision.org/. Accessed 20 Oct 2018

Bengio Y (2017) Mila and the future of Theano. https://groups.google.com/forum/#!msg/theano-users/7Poq8BZutbY/rNCIfvAEAwAJ. Accessed 28 Feb 2018

Bhatia J (2017) Search for the fastest deep learning framework supported by Keras. https://www.datasciencecentral.com/profiles/blogs/search-for-the-fastest-deep-learning-framework-supported-by-keras. Accessed 27 Sept 2018

Bishop CM (2006) Pattern recognition and machine learning (information science and statistics). Springer, Berlin

Braun S (2018) LSTM benchmarks for deep learning frameworks. https://www.groundai.com/project/lstm-benchmarks-for-deep-learning-frameworks/. Accessed 27 Sept 2018

BusinessWire (2017) H2O.ai partners with IBM to bring Enterprise AI to IBM Power Systems. https://www.h2o.ai/company/h2o-ai-partners-with-ibm-to-bring-enterprise-ai-to-ibm-power-systems/. Accessed 14 Sept 2018

Caffe2 (2018) Caffe2—a new lightweight, modular, and scalable deep learning framework. https://caffe2.ai/. Accessed 20 Oct 2018

Caffe2PyTorch (2018) Caffe2 vs PyTorch. https://discuss.pytorch.org/t/caffe2-vs-pytorch/2022/5. Accessed 26 Feb 2018

Cano A (2018) A survey on graphic processing unit computing for large-scale data mining. Wiley Interdiscip Rev Data Min Knowl Discov 8(1):e1232

Cano A, Luna JM, Zafra A, Ventura S (2015) A classification module for genetic programming algorithms in JCLEC. J Mach Learn Res 16(1):491–494

Chainer (2018) Chainer—a powerful, flexible, and intuitive framework for neural networks. https://chainer.org/index.html. Accessed 20 Oct 2018

ChainerMN (2018) ChainerMN : distributed deep learning with chainer. https://github.com/chainer/chainermn. Accessed 04 Oct 2018

Chang CC, Lin CJ (2011) Libsvm: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):27

Chen T, Guestrin C (2016) Xgboost: a scalable tree boosting system. In: Proceedings of the 22ND ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 785–794

Chen T, Li M, Li Y, Lin M, Wang N, Wang M, Xiao T, Xu B, Zhang C, Zhang Z (2015) Mxnet: a flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274

Chintala S (2017) Easy benchmarking of all publicly accessible implementations of convnets. https://github.com/soumith/convnet-benchmarks. Accessed 28 Sept 2018

Clj-ml (2018) A machine learning library for Clojure built on top of Weka and friends. https://github.com/antoniogarrote/clj-ml. Accessed 25 Feb 2018

CNTK (2018) Microsoft cognitive toolkit (CNTK), an open source deep-learning toolkit. https://docs.microsoft.com/en-us/cognitive-toolkit/. Accessed 20 Oct 2018

Codacy (2018) Codacy: automated code reviews and code analytics. https://app.codacy.com/. Accessed 15 Sept 2018

CodeFactor (2018) Codefactor: automated code review for GitHub and Bitbucket. https://www.codefactor.io/. Accessed 17 Sept 2018

Collobert R, Bengio S, Mariethoz J (2002) Torch: a modular machine learning software library (no. epfl-report-82802). Technical report, IDIAP

Collobert R, Kavukcuoglu K, Farabet C (2011) Torch7: a MATLAB-like environment for machine learning. In: BigLearn, NIPS workshop, EPFL-CONF-192376

Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3213–3223. https://www.cityscapes-dataset.com/

Courbariaux M, Hubara I, Soudry D, El-Yaniv R, Bengio Y (2016) Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or −1. arXiv preprint arXiv:1602.02830

Crammer K, Singer Y (2001) On the algorithmic implementation of multiclass kernel-based vector machines. J Mach Learn Res 2:265–292

Crammer K, Singer Y (2002) On the learnability and design of output codes for multiclass problems. Mach Learn 47(2–3):201–233

CRISP-DM (1999) Cross industry standard process for data mining. http://cordis.europa.eu/project/rcn/37679_en.html. Accessed 15 Sept 2018

CUDA (2018) CUDA zone—NVIDIA development. https://developer.nvidia.com/cuda-zone. Accessed 20 Oct 2018

CudaToolkit (2018) NVIDIA CUDA toolkit. https://developer.nvidia.com/cuda-toolkit. Accessed 20 Oct 2018

cuDNN (2018) NVIDIA cuDNN—GPU accelerated deep learning. https://developer.nvidia.com/cudnn. Accessed 20 Oct 2018

Deshpande A (2017) Understanding CNNs part 3. https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html. Accessed 23 Feb 2018

Dieleman S, Willett KW, Dambre J (2015) Rotation-invariant convolutional neural networks for galaxy morphology prediction. Mon Not R Astron Soc 450(2):1441–1459

Digits (2018) Digits. https://developer.nvidia.com/digits. Accessed 28 Feb 2018

DL4J (2018) Deeplearning4j—the first commercial-grade, open-source, distributed deep-learning library written for Java and Scala, integrated with hadoop and spark. https://deeplearning4j.org/. Accessed 22 Sept 2018

DLwiki (2018) Comparison of deep learning software. https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software. Accessed 25 Feb 2018

DMLC (2018) DMLC for scalable and reliable machine learning. http://dmlc.ml/. Accessed 20 Oct 2018

Duch W, Diercksen GH (1994) Neural networks as tools to solve problems in physics and chemistry. Comput Phys Commun 82(2–3):91–103

Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) Liblinear: a library for large linear classification. J Mach Learn Res 9:1871–1874

Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller PA (2018) Deep learning for time series classification: a review. arXiv preprint arXiv:1809.04356

Feldman M (2016) IBM finds killer app for TrueNorth neuromorphic chip. https://www.top500.org/news/ibm-finds-killer-app-for-truenorth-neuromorphic-chip/. Accessed 21 Sept 2018

Felice M (2017) Which deep learning network is best for you. https://www.cio.com/article/3193689/artificial-intelligence/which-deep-learning-network-is-best-for-you.html. Accessed 25 Feb 2018

Flink (2018) Apache Flink: scalable stream and batch data processing. https://flink.apache.org/. Accessed 20 Oct 2018

Giordaniello F, Cognolato M, Graziani M, Gijsberts A, Gregori V, Saetta G, Hager AGM, Tiengo C, Bassetto F, Brugger P et al (2017) Megane pro: myo-electricity, visual and gaze tracking data acquisitions to

improve hand prosthetics. In: 2017 international conference on rehabilitation robotics (ICORR). IEEE, pp 1148–1153
GitHub (2018) GitHub the world's leading software development platform. https://github.com/. Accessed 15 Sept 2018
Gluon (2018) A clear, concise, simple yet powerful and efficient API for deep learning. https://github.com/gluon-api/gluon-api. Accessed 28 Feb 2018
Gonzalez-Lopez J, Ventura S, Cano A (2018) Distributed nearest neighbor classification for large-scale multi-label data on spark. Future Gener Comput Syst 87:66–82
Goodfellow I, Bengio Y, Courville A, Bengio Y (2016) Deep learning, vol 1. MIT Press, Cambridge
Google-AI-Blog (2017) Announcing tensorflow fold: deep learning with dynamic computation graphs. Google AI Blog—the latest new from Google AI. https://ai.googleblog.com/2017/02/announcing-tensorflow-fold-deep.html. Accessed 19 Sept 2018
GoogleTPU (2018) Google announces a new generation for its TPU machine learning hardware. https://techcrunch.com/2018/05/08/google-announces-a-new-generation-for-its-tpu-machine-learning-hardware/?guccounter=1. Accessed 20 Sept 2018
Grafana (2018) Grafana—the open platform for analytics and monitoring. https://grafana.com/. Accessed 25 Feb 2018
H2O (2018) 0xdata—H2O.ai—fast scalable machine learning. http://h2o.ai/. Accessed 20 Oct 2018
H2O.ai (2017) Deep learning (neural networks). http://h2o-release.s3.amazonaws.com/h2o/rel-wheeler/2/docs-website/h2o-docs/data-science/deep-learning.html. Accessed 23 Feb 2018
Hafiane A, Vieyres P, Delbos A (2017) Deep learning with spatiotemporal consistency for nerve segmentation in ultrasound images. arXiv preprint arXiv:1706.05870
Harris M (2016) Mixed-precision programming with CUDA 8. NVidia. https://devblogs.nvidia.com/mixed-precision-programming-cuda-8/. Accessed 18 Sept 2018
He K, Zhang X, Ren S, Sun J (2016a) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
He K, Zhang X, Ren S, Sun J (2016b) Identity mappings in deep residual networks. In: European conference on computer vision. Springer, pp 630–645
Hermans J (2017) On scalable deep learning and parallelizing gradient descent. PhD thesis, Maastricht U
Hickey R (2018) The Clojure programming language. https://clojure.org/. Accessed 14 Sept 2018
Huang G, Liu Z, Weinberger KQ, van der Maaten L (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 1, p 3
Hwang JJ, Azernikov S, Efros AA, Yu SX (2018) Learning beyond human expertise with generative models for dental restorations. arXiv preprint arXiv:1804.00064
Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and less than 0.5 mb model size. arXiv preprint arXiv:1602.07360
IMDb (2018) IMDb datasets. https://www.imdb.com/interfaces/. Accessed 28 Sept 2018
Jean N, Burke M, Xie M, Davis WM, Lobell DB, Ermon S (2016) Combining satellite imagery and machine learning to predict poverty. Science 353(6301):790–794
Jolav (2018) GitHub star history. https://codetabs.com/github-stars/github-star-history.html. Accessed 15 Sept 2018
Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R, Bates S, Bhatia S, Boden N, Borchers A et al (2017) In-datacenter performance analysis of a tensor processing unit. In: 2017 ACM/IEEE 44th annual international symposium on computer architecture (ISCA). IEEE, pp 1–12
Jovic A, Brkic K, Bogunovic N (2014) An overview of free software tools for general data mining. In: 2014 37th international convention on information and communication technology, electronics and microelectronics (MIPRO). IEEE, pp 1112–1117
Jupyter (2018) Project jupyter. https://jupyter.org/. Accessed 25 Feb 2018
Kalogeiton V, Lathuiliere S, Luc P, Lucas T, Shmelkov K (2016) Deep learning frameworks: Tensorflow, Theano, Keras, Torch and Caffe. https://project.inria.fr/deeplearning/files/2016/05/DLFrameworks.pdf. Accessed 23 Oct 2018
Kalray (2017) Deep learning for high-performance applications. http://www.eenewseurope.com/Learning-center/kalray-deep-learning-high-performance-applications. Accessed 23 Feb 2018
Karmanov I, Salvaris M, Fierro M, Dean D (2018) Comparing deep learning frameworks: a Rosetta stone approach. https://github.com/ilkarman/DeepLearningFrameworks. Accessed 28 Sept 2018
Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L (2014) Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1725–1732
Keras (2018) Keras—high-level neural networks API. https://keras.io/. Accessed 20 Oct 2018

Khronos (2018) OpenCL—open computing language. The Khronos Group Inc. https://www.khronos.org/opencl/. Accessed 15 Sept 2018
Kibana (2018) Kibana: explore, visualize, discover data—elastic. https://www.elastic.co/products/kibana. Accessed 25 Feb 2018
Kloss C (2017) Intel Nervana neural network processor: architecture update. https://ai.intel.com/intel-nervana-neural-network-processor-architecture-update/. Accessed 21 Sept 2018
KNIME (2018) KNIME—open for innovation. https://www.knime.com/. Accessed 25 Feb 2018
Kobielus J (2018) Powering AI: the explosion of new AI hardware accelerators. https://www.infoworld.com/article/3290104/artificial-intelligence/powering-ai-the-explosion-of-new-ai-hardware-accelerators.html. Accessed 14 Sept 2018
Konsor P (2012) Intel software—developer zone—performance benefits of half precision floats, Intel Software Development Zone. https://software.intel.com/en-us/articles/performance-benefits-of-half-precision-floats. Accessed 23 Feb 2018
Krizhevsky A (2009) CIFAR datasets. http://www.cs.toronto.edu/~kriz/cifar.html. Accessed 28 Sept 2018
Krizhevsky A (2014) One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997
Lacey G, Taylor GW, Areibi S (2016) Deep learning on FPGAs: past, present, and future. arXiv preprint arXiv:1602.04283
Lasagne (2018) Lightweight library to build and train neural networks in Theano. https://github.com/Lasagne/Lasagne. Accessed 28 Feb 2018
LeCun (1998) The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/. Accessed 28 Sept 2018
LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken AP, Tejani A, Totz J, Wang Z et al (2017) Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR, vol 2, p 4
Lee J (2017) Keras backend benchmark: Theano vs Tensorflow vs CNTK. http://kaggler.com/keras-backend-benchmark-theano-vs-tensorflow-vs-cntk/. Accessed 27 Sept 2018
LibLinear (2018) Liblinear—a library for large linear classification. https://www.csie.ntu.edu.tw/~cjlin/liblinear/. Accessed 20 Oct 2018
LibSVM (2018) LibSVM—a library for support vector machines. https://www.csie.ntu.edu.tw/~cjlin/libsvm/. Accessed 20 Oct 2018
Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context. In: European conference on computer vision, Springer, pp 740–755. http://cocodataset.org/#home
LISA (2015) Deep learning tutorial. University of Montreal, LISA Lab
Liu J, Li J, Li W, Wu J (2016) Rethinking big data: a review on the data quality and usage issues. ISPRS J Photogramm Remote Sens 115:134–142
Liu J, Dutta J, Li N, Kurup U, Shah M (2018) Usability study of distributed deep learning frameworks for convolutional neural networks. KDD. http://www.kdd.org/kdd2018/files/deep-learning-day/DLDay18_paper_29.pdf
Markidis S, Der Chien SW, Laure E, Peng IB, Vetter JS (2018) NVidia tensor core programmability, performance & precision. arXiv preprint arXiv:1803.04014
MatLab (2018) Matlab—the language of technical computing. http://www.mathworks.com/products/matlab/. Accessed 25 Feb 2018
Microsoft (2017) Microsoft unveils project brainwave for real-time AI. https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/. Accessed 21 Sept 2018
Mierswa I (2017) What is artificial intelligence, machine learning, and deep learning. https://rapidminer.com/artificial-intelligence-machine-learning-deep-learning/. Accessed 22 Feb 2018
Mierswa I, Klinkenberg R, Fischer S, Ritthoff O (2003) A flexible platform for knowledge discovery experiments: Yale—yet another learning environment. In: LLWA 03-Tagungsband der GI-Workshop-Woche Lernen-Lehren-Wissen-Adaptivität
Migdal P, Jakubanis R (2018) Keras vs pytorch: Keras or Pytorch as your first deep learning framework. https://deepsense.ai/keras-or-pytorch/. Accessed 2 Oct 2018
MILA (2018) Montreal Institute for Learning Algorithms. http://mila.umontreal.ca/. Accessed 28 Feb 2018
MIOpen (2018) Miopen—AMD's ML library. https://github.com/ROCmSoftwarePlatform/MIOpen. Accessed 25 Sept 2018
Mitchell R (2017) Gradient boosting, decision trees and XGBoost with CUDA. https://devblogs.nvidia.com/parallelforall/gradient-boosting-decision-trees-xgboost-cuda/. Accessed 25 Feb 2018

MKL (2018) Intel MKL—Intel Math Kernel Library. https://software.intel.com/en-us/intel-mkl/. Accessed 23 Feb 2018
Mnih V (2013) Machine learning for aerial image labeling. University of Toronto (Canada)
Mustafa M, Bard D, Bhimji W, Al-Rfou R, Lukić Z (2017) Creating virtual universes using generative adversarial networks. arXiv preprint arXiv:1706.02390
MXNet (2018) Apache MXNet—a flexible and efficient library for deep learning. https://mxnet.apache.org/. Accessed 20 Oct 2018
Nguyen G, Nguyen BM, Tran D, Hluchy L (2018) A heuristics approach to mine behavioural data logs in mobile malware detection system. Data Knowl Eng 115:129–151
NLTK (2018) Natural language toolkit. http://www.nltk.org/. Accessed 25 Feb 2018
NTU (2018) Distributed LibLinear: libraries for large-scale linear classification on distributed environments. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/distributed-liblinear/. Accessed 3 Oct 2018
NumPy (2018) NumPy—the fundamental package for scientific computing with Python. http://www.numpy.org/. Accessed 25 Feb 2018
NVIDIA (2018) NVIDIA accelerated computing. https://developer.nvidia.com/computeworks. Accessed 20 Oct 2018
Octave (2018) GNU Octave scientific programming language. https://www.gnu.org/software/octave/. Accessed 25 Feb 2018
OpenMP (2018) OpenMP—API specification for parallel programming. www.openmp.org. Accessed 23 Feb 2018
OpenMPI (2018) OpenMPI—open source high performance computing. https://www.open-mpi.org/. Accessed 23 Feb 2018
Oryx2 (2018) Oryx2—framework for real-time large scale machine learning. http://oryx.io/. Accessed 20 Oct 2018
Otalora S, Schaer R, Atzori M, del Toro OAJ, Muller H (2018) Deep learning based retrieval system for gigapixel histopathology cases and open access literature. bioRxiv 408237
Paganini M, de Oliveira L, Nachman B (2018) Calogan: simulating 3d high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. Phys Rev D 97(1):014021
Pandas (2018) Pandas—Python Data Analysis Library. https://pandas.pydata.org/. Accessed 25 Feb 2018
Patel M (2017) When two trends fuse: Pytorch and recommender systems. O'Reilly Media. https://www.oreilly.com/ideas/when-two-trends-fuse-pytorch-and-recommender-systems. Accessed 19 Sept 2018
Patel H (2018) Tensorflow pros and cons—the bright and the dark sides. https://medium.com/@patelharshali136/tensorflow-pros-and-cons-the-bright-and-the-dark-sides-a7cc2388de8b. Accessed 26 Feb 2018
Piatetsky G (2017) Python vs R. https://www.kdnuggets.com/2017/09/python-vs-r-data-science-machine-learning.html. Accessed 25 Feb 2018
Pol AA, Cerminara G, Germain C, Pierini M, Seth A (2018) Detector monitoring with artificial neural networks at the CMS experiment at the CERN Large Hadron Collider. arXiv preprint arXiv:1808.00911
PSPP (2018) GNU PSPP for statistical analysis of sampled data. https://www.gnu.org/software/pspp/. Accessed 25 Feb 2018
Python (2018) Python programming language. https://www.python.org/. Accessed 25 Feb 2018
PyTorch (2018) PyTorch—deep learning framework that puts python first. http://pytorch.org/. Accessed 20 Oct 2018
PyTorchTeam (2018) The road to 1.0: production ready PyTorch. https://pytorch.org/blog/the-road-to-1_0/. Accessed 19 Sept 2018
R-CRAN (2018) Comprehensive R Archive Network (cran). https://cran.r-project.org/. Accessed 25 Feb 2018
Rajpurkar P, Irvin J, Zhu K, Yang B, Mehta H, Duan T, Ding D, Bagul A, Langlotz C, Shpanskaya K et al (2017) Chexnet: radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225
Ramirez-Gallego S, Krawczyk B, Garcia S, Wozniak M, Benitez JM, Herrera F (2017) Nearest neighbor classification for high-speed big data streams using spark. IEEE Trans Syst Man Cybern Syst 47(10):2727–2739
Rapid (2018) RapidMiner open source predictive analytics platform. https://rapidminer.com/. Accessed 25 Feb 2018
RapidMiner (2013) The core of RapidMiner is open source. https://rapidminer.com/blog/the-core-of-rapidminer-is-open-source/. Accessed 22 Sept 2018
Rehurek R (2018) Gensim topic modelling for human. https://radimrehurek.com/gensim/. Accessed 17 May 2018
ROCm (2016) ROCm, a new era in open GPU computing. https://rocm.github.io/. Accessed 25 Sept 2018
ROCm-DL (2018) Deep learning framework support for ROCm. https://rocm-documentation.readthedocs.io/en/latest/Deep_learning/Deep-learning.html. Accessed 25 Sept 2018

ROCm-HIP (2018) Hip: Convert CUDA to portable C++ code. https://github.com/ROCm-Developer-Tools/HIP. Accessed 25 Sept 2018
Rouse M (2018) Intelligent system. https://whatis.techtarget.com/definition/intelligent-system. Accessed 15 Sept 2018
Rproject (2018) R project for statistical computing. http://www.r-project.org/. Accessed 25 Feb 2018
Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
Sam DB, Surya S, Babu RV (2017) Switching convolutional neural network for crowd counting. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 1, p 6
SAS (2018) SAS (previously statistical analysis system). https://www.sas.com/en_us/. Accessed 25 Feb 2018
Schlegl T, Seeböck P, Waldstein SM, Schmidt-Erfurth U, Langs G (2017) Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: International conference on information processing in medical imaging. Springer, pp 146–157
Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
Scikit (2018) Scikit-learn machine learning in Python. http://scikit-learn.org/stable/. Accessed 20 Oct 2018
SciLab (2018) SciLab—open source software for numerical computation. https://www.scilab.org/. Accessed 25 Feb 2018
SciPy (2018) SciPy—Python-based ecosystem of open-source software for mathematics, science, and engineering. https://www.scipy.org/. Accessed 25 Feb 2018
Shi S, Wang Q, Xu P, Chu X (2016) Benchmarking state-of-the-art deep learning software tools. In: 2016 7th international conference on cloud computing and big data (CCBD). IEEE, pp 99–104
Shogun (2018) Shogun official web page. http://www.shogun.ml/. Accessed 24 Feb 2018
ShogunGoogle (2018) Shogun machine learning toolbox—Google summer of code archive. https://summerofcode.withgoogle.com/archive/2017/organizations/4704476053110784/. Accessed 24 Feb 2018
Siam M, Elkerdawy S, Jagersand M, Yogamani S (2017) Deep semantic segmentation for automated driving: taxonomy, roadmap and challenges. In: 2017 IEEE 20th international conference on intelligent transportation systems (ITSC). IEEE, pp 1–8
Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
Skymind (2017) Comparing top deep learning frameworks: Deeplearning4j, Pytorch, Tensorflow, Caffe, Keras, MXNet, Gluon and CNTK. https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch. Accessed 28 Feb 2018
Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, pp 2951–2959
Sonnenburg S, Henschel S et al (2010) The shogun machine learning toolbox. Journal of Machine Learning Research 11(Jun):1799–1802
Sonnet (2018) Sonnet (Deepmind). https://github.com/deepmind/sonnet. Accessed 28 Feb 2018
Spark (2018a) Announcement: dataframe-based API is primary API. https://spark.apache.org/docs/latest/ml-guide.html. Accessed 17 Sept 2018
Spark (2018b) Apache Spark—fast and general engine for large-scale data processing. https://spark.apache.org/. Accessed 22 Sept 2018
SPSS (2018) SPSS. http://www.ibm.com/software/analytics/spss/. Accessed 25 Feb 2018
Su P, Ding X, Zhang Y, Li Y, Zhao N (2017) Predicting blood pressure with deep bidirectional LSTM network. arxiv preprint. arXiv preprint arXiv:1705.04524
Sze V, Chen YH, Yang TJ, Emer JS (2017) Efficient processing of deep neural networks: a tutorial and survey. Proc IEEE 105(12):2295–2329
Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
Tableau (2018) Tableau software: business intelligence and analytics. https://www.tableau.com/. Accessed 25 Feb 2018
Tang S (2018) AI/ML/DL ICs and IPs. https://github.com/basicmi/Deep-Learning-Processor-List. Accessed 14 Sept 2018
Tensor P (2018) Pretty tensor. https://github.com/google/prettytensor. Accessed 28 Feb 2018
TensorFlow (2018) TensorFlow—an open-source software library for machine intelligence. https://www.tensorflow.org/. Accessed 20 Oct 2018
TensorFlowCommunity (2018) TensorFlow—roadmap. https://www.tensorflow.org/community/roadmap. Accessed 20 Sept 2018
TensorFlowLite (2018) TensorFlowLite. https://www.tensorflow.org/mobile/. Accessed 20 Oct 2018

TensorLayer (2018) TensorLayer. https://tensorlayer.readthedocs.io/en/latest/. Accessed 28 Feb 2018

TFLean (2018) TFLean. http://tflearn.org/. Accessed 28 Feb 2018

TFSlim (2018) TF-Slim–Tensorflow-Slim—lightweight library for defining, training and evaluating complex models in Tensorflow. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim. Accessed 28 Feb 2018

Theano (2018) Theano. http://deeplearning.net/software/theano. Accessed 20 Oct 2018

Tokui S, Oono K, Hido S, Clayton J (2015) Chainer: a next-generation open source framework for deep learning. In: Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS), vol 5, pp 1–6

Torch (2018) Torch—scientific computing framework for LuaJIT. http://torch.ch/. Accessed 26 Feb 2018

Torro FB, Quilis JDS, Espert IB, Bayarri AA, Bonmati LM (2017) Accelerating the diffusion-weighted imaging biomarker in the clinical practice: comparative study. Proc Comput Sci 108:1185–1194

Triguero I, Gonzalez S, Moyano JM, Garcia S, Alcala-Fdez J, Luengo J, Fernandez A, del Jesus MJ, Sanchez L, Herrera F (2017) Keel 3.0: an open source software for multi-stage analysis in data mining. Int J Comput Intell Syst 10(1):1238–1249

Trippi RR, Turban E (eds) (1992) Neural networks in finance and investing: using artificial intelligence to improve real world performance. McGraw-Hill, New York

Van Horn G, Mac Aodha O, Song Y, Cui Y, Sun C, Shepard A, Adam H, Perona P, Belongie S (2018) The iNaturalist species classification and detection dataset. thecvf.com

Varangaonkar A (2017) Top 10 deep learning frameworks. https://datahub.packtpub.com/deep-learning/top-10-deep-learning-frameworks. Accessed 22 Sept 2018

Veen FV (2016) The neural network zoo. http://www.asimovinstitute.org/neural-network-zoo/. Accessed 22 Feb 2018

Vryniotis V (2018) 5 tips for multi-GPU training with Keras. http://blog.datumbox.com/5-tips-for-multi-gpu-training-with-keras/. Accessed 19 Sept 2018

VW (2018) Vowpal Wabbit open source fast learning system. https://github.com/JohnLangford/vowpal_wabbit/wiki. Accessed 20 Oct 2018

VWAzure (2018) Text analytics and Vowpal Wabbit in Azure Machine Learning Studio. https://azure.microsoft.com/en-in/documentation/videos/text-analytics-and-vowpal-wabbit-in-azure-ml-studio/. Accessed 25 Feb 2018

Waikato (2018) Read-only mirror of the official Weka subversion repository (3.8.x). https://github.com/Waikato/weka-3.8. Accessed 17 Sept 2018

Wainberg M, Merico D, Delong A, Frey BJ (2018) Deep learning in biomedicine. Nat Biotechnol 36(9):829

Wang TC, Liu MY, Zhu JY, Tao A, Kautz J, Catanzaro B (2018) High-resolution image synthesis and semantic manipulation with conditional GANs. In: IEEE conference on computer vision and pattern recognition (CVPR), vol 1, p 5

Weinberger K, Dasgupta A, Langford J, Smola A, Attenberg J (2009) Feature hashing for large scale multitask learning. In: Proceedings of the 26th annual international conference on machine learning. ACM, pp 1113–1120

Weka3 (2018) Weka3: data mining software in Java. http://www.cs.waikato.ac.nz/ml/weka/. Accessed 25 Feb 2018

Wielgosz M, Skoczeń A, Mertik M (2017) Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets. Nucl Instrum Methods Phys Res A 867:40–50. https://doi.org/10.1016/j.nima.2017.06.020

Woolf M (2017) Benchmarking CNTK on Keras: is it better at deep learning than tensorflow? https://www.datasciencecentral.com/profiles/blogs/search-for-the-fastest-deep-learning-framework-supported-by-keras. Accessed 27 Sept 2018

Wu J, Zhang C, Xue T, Freeman B, Tenenbaum J (2016a) Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in neural information processing systems, pp 82–90

Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K et al (2016b) Google's neural machine translation system: bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144

Xu J, Gong E, Pauly J, Zaharchuk G (2017) 200x low-dose pet reconstruction using deep learning. arXiv preprint arXiv:1712.04119

Yang LC, Chou SY, Yang YH (2017) Midinet: a convolutional generative adversarial network for symbolic-domain music generation. arXiv preprint arXiv:1703.10847

Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag 13(3):55–75

Zagoruyko S, Komodakis N (2016) Wide residual networks. arXiv preprint arXiv:1605.07146

Zamecnik B (2017) Towards efficient multi-gpu training in Keras with Tensorflow. https://medium.com/rossum/towards-efficient-multi-gpu-training-in-keras-with-tensorflow-8a0091074fb2. Accessed 19 Sept 2018
Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer, pp 818–833
Zeppelin (2018) Apache Zeppelin. https://zeppelin.apache.org/. Accessed 25 Feb 2018
Zhang H, Xu T, Li H, Zhang S, Huang X, Wang X, Metaxas D (2017) StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. arXiv preprint
Zhu JY, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. arXiv preprint
Zygmunt Z (2014) Vowpal Wabbit, LibLinear/SBM and StreamSVM compared. http://fastml.com/vowpal-wabbit-liblinear-sbm-and-streamsvm-compared/. Accessed 20 Oct 2018

Affiliations

Giang Nguyen[1] · Stefan Dlugolinsky[1] · Martin Bobák[1] · Viet Tran[1] · Álvaro López García[2] · Ignacio Heredia[2] · Peter Malík[1] · Ladislav Hluchý[1]

Stefan Dlugolinsky
stefan.dlugolinsky@savba.sk

Martin Bobák
martin.bobak@savba.sk

Viet Tran
viet.tran@savba.sk

Álvaro López García
aloga@ifca.unican.es

Ignacio Heredia
iheredia@ifca.unican.es

Peter Malík
p.malik@savba.sk

Ladislav Hluchý
hluchy.ui@savba.sk

1 Institute of Informatics, Slovak Academy of Science (IISAS), Dubravska cesta 9, 845 07 Bratislava, Slovakia

2 Instituto de Fisica de Cantabria (IFCA - CSIC), Ed. Juan Jorda, Avda. de los Castros s/n, 39005 Santander, Spain