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

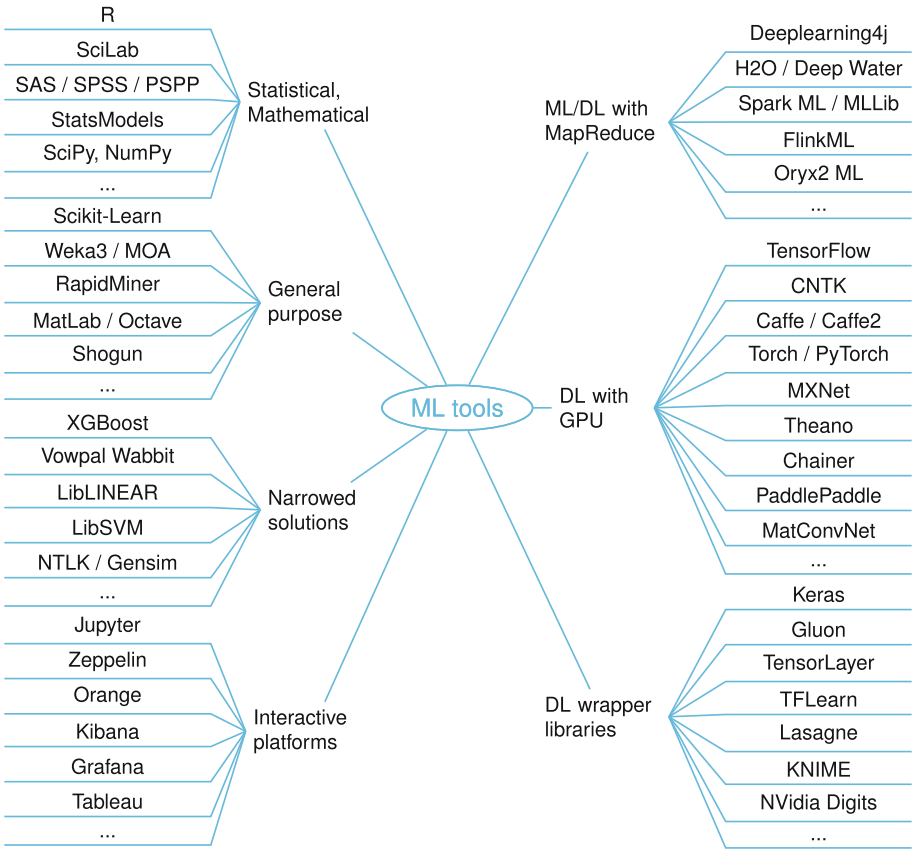

Fig. 2 Overview of Machine Learning frameworks and libraries

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

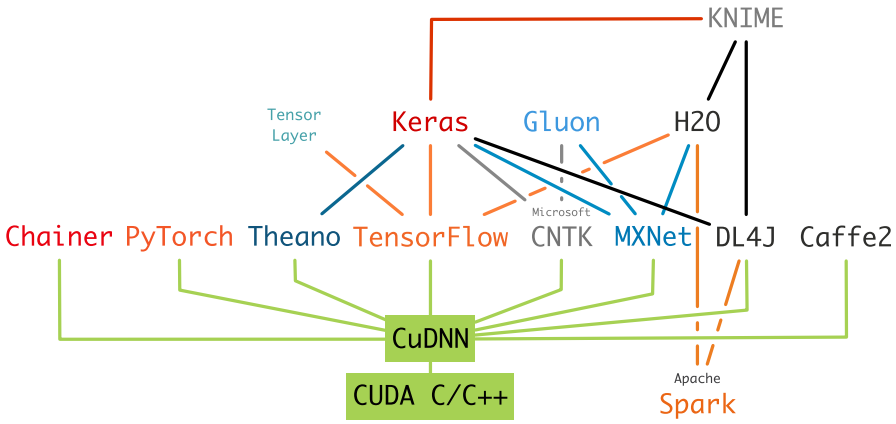

Fig. 3 The most popular Deep Learning frameworks and libraries layering in various abstraction implementation levels

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 × 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 Intelligence 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 preprocessing 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.