

# DEEP-Hybrid-DataCloud

## STATE-OF-THE-ART DEEP LEARNING (DL), NEURAL NETWORK (NN) AND MACHINE LEARNING (ML) FRAMEWORKS AND LIBRARIES

**DELIVERABLE: D6.1**

---

**Document identifier:** DEEP-JRA3-D6.1

**Date:** 28/02/2018

**Activity:** WP6

**Lead partner:** IISAS

**Status:** FINAL

**Dissemination level:** PUBLIC

**Permalink:** <https://confluence.deep-hybrid-datacloud.eu/download/attachments/3145850/DEEP-JRA3-D6.1.pdf>

---

### Abstract

This document provides an overview of the state-of-the-art in Deep Learning (DL), Neural Network (NN) and Machine Learning (ML) frameworks and libraries to be used as building blocks in the DEEP Open Catalogue. The initial state of the catalogue will be built based on the outcome of this document and the initial user community requirements of scientific data analytic and ML/DL tools coming from WP2.

## Copyright Notice

Copyright © Members of the DEEP Hybrid-DataCloud Collaboration, 2017-2020.

## Delivery Slip

	Name	Partner/Activity	Date
<b>From</b>	Giang Nguyen	IISAS/JRA3	28/02/2018
<b>Reviewed by</b>	Ignacio Heredia Elisabetta Ronchieri Ignacio Blanquer Álvaro López García	CSIC INFN UPV CSIC	16/02/2018
<b>Approved by</b>	Steering Committee		22/02/2018

## Document Log

Issue	Date	Comment	Author/Partner
TOC	31/12/2017	Table of content	Giang Nguyen / IISAS
Draft	31/01/2018	First draft version	Giang Nguyen / IISAS
Review	14/02/2018	Review version	Giang Nguyen / IISAS
Final	28/02/2018	Final version	Giang Nguyen / IISAS Álvaro López / CSIC

## Table of Contents

Executive Summary.....	5
1. Introduction.....	6
2. Machine Learning and Deep Learning at a glance.....	8
2.1. Machine Learning approach.....	9
2.2. From Neural Networks to Deep Learning.....	12
2.2.1. Deep Neural Networks and Deep Learning architectures.....	12
2.2.2. Deep Learning timeline through the most well-known models.....	15
2.2.3. Problems in Deep Learning and advanced algorithmic solutions.....	15
2.3. Accelerated computing and Deep Learning.....	16
2.3.1. Accelerated libraries.....	17
2.3.2. Digital ecosystems and the embedding trend.....	19
3. State-of-the-art of Machine Learning frameworks and libraries.....	19
3.1. General Machine Learning frameworks and libraries.....	21
3.1.1. Shogun.....	24
3.1.2. RapidMiner.....	24
3.1.3. Weka3.....	25
3.1.4. Scikit-Learn.....	26
3.1.5. LibSVM.....	26
3.1.6. LibLinear.....	27
3.1.7. Vowpal Wabbit.....	28
3.1.8. XGBoost.....	28
3.1.9. Interactive data analytics and data visualisation.....	29
3.1.10. Other tools including data analytic frameworks and libraries.....	30
3.2. Deep Learning frameworks and libraries with GPU support.....	31
3.2.1. TensorFlow.....	34
3.2.2. Keras.....	35
3.2.3. CNTK.....	35
3.2.4. Caffe.....	36
3.2.5. Caffe2.....	37
3.2.6. Torch.....	37
3.2.7. PyTorch.....	38
3.2.8. MXNet.....	39
3.2.9. Theano.....	40
3.2.10. Chainer.....	40
3.2.11. Wrapper frameworks and libraries.....	41
3.2.12. Other DL frameworks and libraries with GPU supports.....	42
3.3. Machine Learning and Deep Learning frameworks and libraries with MapReduce.....	44
3.3.1. Deeplearning4j.....	46
3.3.2. Apache Spark MLlib and ML.....	46
3.3.3. H2O, Sparkling and Deep Water.....	48
3.3.4. Other frameworks and libraries coupled with MapReduce.....	49
4. Conclusions.....	50
5. References.....	52
5.1. Links.....	56
6. Glossary.....	62
6.1. List of Figures.....	62
6.2. List of Tables.....	62
6.3. Acronyms.....	63



## Executive Summary

The DEEP-HybridDataCloud (Designing and Enabling E-Infrastructures for intensive data Processing in a Hybrid DataCloud) is a project approved in July 2017 within the EINFRA-21-2017 call of the Horizon 2020 framework program of the European Community. It will develop innovative services to support intensive computing techniques that require specialized HPC hardware, such as GPUs or low-latency interconnects, to explore very large datasets.

Although the cloud model offers flexibility and scalability, it is quite complex for a scientific researcher that develops a new application to use and exploit the required services at different layers. Within the project, WP6 is going to realize the DEEP as a Service solution composed of a set building blocks (i.e. the DEEP Open Catalogue) that enable the easy development of compute-intensive applications. By using this solution users will get easy access to cutting-edge computing libraries (such as deep learning and other compute-intensive techniques) adapted to leverage high-end accelerators (GPUs), integrated with BigData analytics frameworks existing in other initiatives and e-Infrastructures (like the EOSC or EGI.eu). The DEEP as a Service solution will therefore lower the access barrier for scientists, fostering the adoption of advanced computing techniques, large-scale analysis and post-processing of existing data.

This deliverable provides the state-of-the-art in Deep Learning (DL), Neural Network (NN) and Machine Learning (ML) frameworks and libraries to be used as building blocks in the DEEP Open Catalogue. It also presents a comprehensive knowledge background about ML and DL for large-scale data mining. The deliverable states clearly the recent time-slide of ML/DL research as well as the current high dynamic development of cutting-edge DL/NN/ML software. By combining one or more of the building blocks, users will be able to describe their application requirements. The DEEP Open Catalogue will be oriented to cover divergent needs and requirements of worldwide researchers and data scientists supported by specialised hardware and the recent current-edge work on compute- and data-intensive libraries and frameworks in the era of large-scale data processing and data mining.

The initial establishment towards scientific data analytic and ML/DL tools will be built based on the outcome of this document and the initial requirements from DEEP research community coming from WP2 (Deliverable D2.1). The content of the DEEP Open Catalogue will be extendable and modifiable according to user community demands.

# 1. Introduction

Nowadays, the full *CRISP-DM* (Cross-Industry Standard Process for Data Mining) cycle is applied in real life data mining (DM) applications with machine learning (ML) techniques. The realization of DM in many life areas, as described also in the CRISP-DM cycle, leads to the need of various tools for statistics, data analytics, data processing, data mining, modelling and evaluation. The CRISP-DM was involved in the EU FP4-ESPRIT 4, ID 24959 project [CRIPS-DM 1999], which is in-part funded by the European Commission. It is now the leading and a *de facto* standard for DM applications.

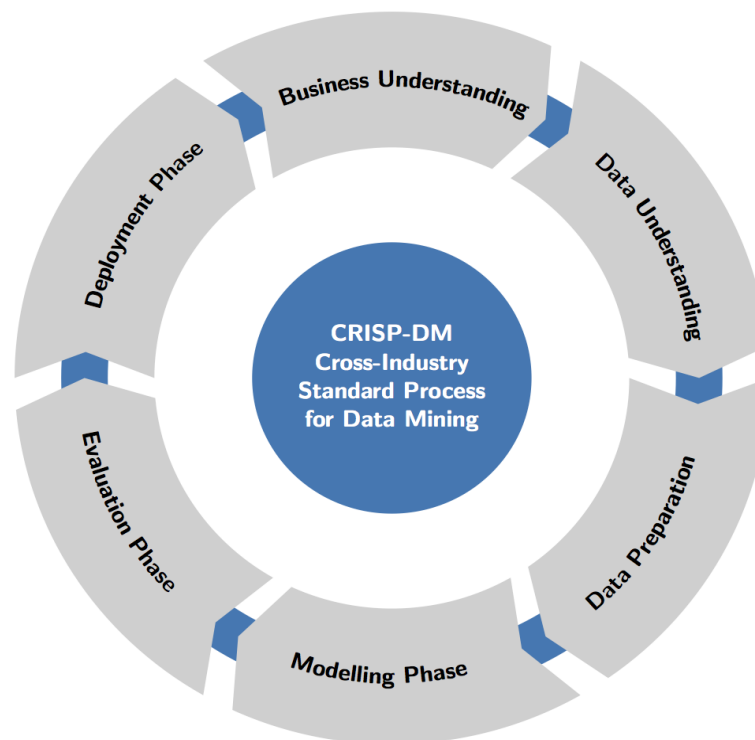


Fig. 1 CRISP-DM Cross-Industry Standard Process for Data Mining

The CRISP-DM consists of six steps: business understanding, data understanding, data preparation, modelling, evaluation and deployment (Fig. 1).

- The business understanding is usually realised based on the provided quest formulations and data descriptions.
- The data understanding is usually realised based on provided data and their documentations.
- The data preparation consists of data transformation, exploratory data analysis and feature engineering, each of them are furthermore divided into smaller sub-steps.
- 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 the model train-test-evaluation cycle. If the data is large-scale, the modelling phase can have time-consuming and computation-intensive requirements.

- The evaluation phase can be performed under various criteria for thorough testing of ML models and to choose the best model for the deployment phase.
- The deployment phase is also called the production phase; it involves the use of the selected ML model as well as the creation of a data pipeline in production.

The whole CRISP-DM cycle is repetitive. The group of the first five phases are also called the development and it can be repeated with different settings according to evaluation results. Here there is the need to highlight the fact that ML algorithms learn from data. Therefore, in practice, data understanding and data preparation phases can consume a large portion of the entire time of every DM project using ML techniques.

Recently, almost all disciplines and research areas, including computer science, business, and medicine, are deeply involved in this spreading computational culture of Big Data because of its broad reach of influence and potential within multiple disciplines. The change in data collection has led to changes in data processing. The Big Data definition is characterised by many Vs, such as Volume, Velocity and Variety, as well as Veracity, Variability, Visualisation, Value and so on. Consequently, the methods and procedures to process these large-scale data must have the capability to handle, e.g., high volume and real-time data. Furthermore, data analysis is expected to change in this new era. The feature of large-scale data requires new approaches and new tools that can accommodate them with different data structures, different spatial and temporal scales [Liu 2016]. The surge of large volumes of information, especially with the Variety characteristic in the Big Data era, to be processed by DM and ML algorithms demand *new transformative parallel and distributed computing solutions* capable to scale computation effectively and efficiently. Graphic processing units (GPUs) have become widespread tools for speeding up general purpose computation in the last decade [Cano 2017]. They offer a massive parallelism to extend algorithms to large-scale data for a fraction of the cost of a traditional high-performance CPU cluster.

The content of the document is organised as follows. Part 1 gives an introduction to data mining for large-scale data. Part 2 presents a comprehensive overview, the evolution and the emerging trend in ML and DL. It also briefly describes the connection between DL and accelerated computing. The main part of the document is Part 3, which provides the state-of-the-art in DL, NN and ML frameworks and libraries. This part is divided into three subparts: general frameworks and libraries, DL with GPU support, and ML/DL integrated with MapReduce. Finally, Part 4 concludes the document.

## 2. Machine Learning and Deep Learning at a glance

**Data Mining** (DM) is the core stage of the knowledge discovery process that is aim to extract interesting and potentially useful information from data [Goodfellow 2016] [Mierswa 2017]. As the DM techniques and businesses evolved, there is a need for data analysts to better understand and standardise the knowledge discovery process. DM can serve as a foundation for Artificial Intelligence and Machine Learning. The term "data mining", as described in this document, is meanly oriented to large-scale data mining. However, many techniques that work for large-scale datasets can work also for small data.

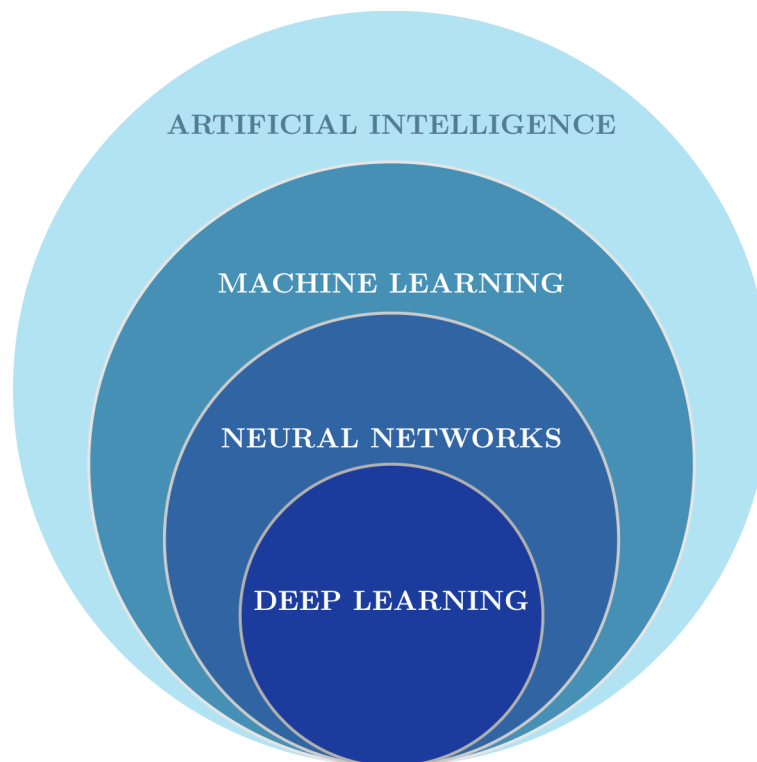


Fig. 2 Relationship between Artificial Intelligence, Machine Learning, Neural Networks and Deep Learning

**Artificial Intelligence** (AI) is any technique that enables computers to mimic human behaviour, including machine learning, Natural Language Processing (NLP), language synthesis, computer vision, robotics, sensor analysis, optimization and simulation, and many more.

**Machine Learning** (ML) is a subset of AI techniques that enables computer systems to learn from previous experience and improve their behaviour. 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 Neural Networks (ANNs) are a subset of ML techniques, which are loosely inspired by biological neural networks included Deep Learning. They are usually described as a collection of connected units, called artificial neurons and organised 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 (GANs), deep belief networks (DBNs), and many more.

The relations among Artificial Intelligence, Machine Learning, Neural Networks and Deep Learning are depicted in Fig. 2.

## 2.1. Machine Learning approach

When facing the 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) the size, quality, and nature of the domain data; 2) the available computational time; 3) the urgency of the task and 4) what is the aim of the quest.

In many cases, no one can tell which algorithm will perform the best before trying different algorithms after thoughtful data examination. The use of a concrete algorithm is usually chosen based on data characteristics and exploratory data analysis. As in general with DM using ML approach, the performance of data models is strongly dependent on the representativeness of the provided data set. The complementarity of methods leads to try different options 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 potential of the data can be tapped by a cooperation of partial weak models e.g. using ensemble learning methods based on principles such as voting, record weighting, multiple training process or random selection. 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.

The simplest customary way is to categorize ML algorithms into supervised, unsupervised and semi-supervised learning [Goodfellow 2016] as follows.

- **Supervised learning** algorithms are learning algorithms that infer a function from some inputs with some outputs using supervised training data that consist of a set of training examples. Each example is a pair of an input and a (desired) output value. In many cases, the output may be difficult to collect automatically and must be provided by a human supervisor (i.e. labeling). The inferred function is called a classifier (if the output is discrete) or a regression function (if the output is continuous).
- **Unsupervised learning** attempts to extract information from a training data that is only based on a set of inputs (i.e. without labeling). This category is usually associated with density estimation, learning to draw samples from a distribution, learning to denoise data from some distribution, finding a manifold that the data lies near, or clustering the data into groups of related examples. The distinction between supervised and unsupervised

algorithms is not formally and rigidly defined because there is no objective test for distinguishing whether a value is a feature or a target provided by a supervisor.

- **Semi-supervised learning** tries to make use of unlabeled data for training e.g. typically from small amount of labeled data within a large amount of unlabeled data. These algorithms are halfway between supervised and unsupervised learning. The reason is the expensive cost associated with the labeling process, e.g. by human expert interventions or physical examinations that causes fully labeled training set infeasible. Semi-supervised learning is interesting from ML theoretical side as a model of human learning.

It is interesting to notice that ML algorithms have no strict categorization, e.g. some method can be listed in one or more categories. For example, NNs can be trained for some problems in a supervised manner while in other problems in an unsupervised manner. Although the problem of algorithm categorization is interesting, it is out of the scope of this document.

**Pre-processing and post-processing algorithms** can also be categorized into a number of sub-categories such as dimensionality reduction, sampling (subsampling, oversampling), linear methods, statistical testing, feature engineering with feature extraction, feature encoding, feature transformation and feature selection (e.g. mutual information, chi-square  $\chi^2$  statistics). Many more algorithms can be listed here for overfitting prevention (e.g. regularization, threshold setting, pruning, dropout), model selection and performance optimization (e.g. hyper-parameter tuning, grid search, local minimum search, bio-inspired optimization) and model evaluation (e.g. 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 (ROC AUC), mean absolute error (MAE), mean squared error (MSE), and root-mean-square error (RMSE).

Fig. 3 provides a comprehensive graphical overview of ML methods for modelling as well as for pre-processing and post-processing. However, this overview is the subject to change as the number of ML algorithms is increasing continually.

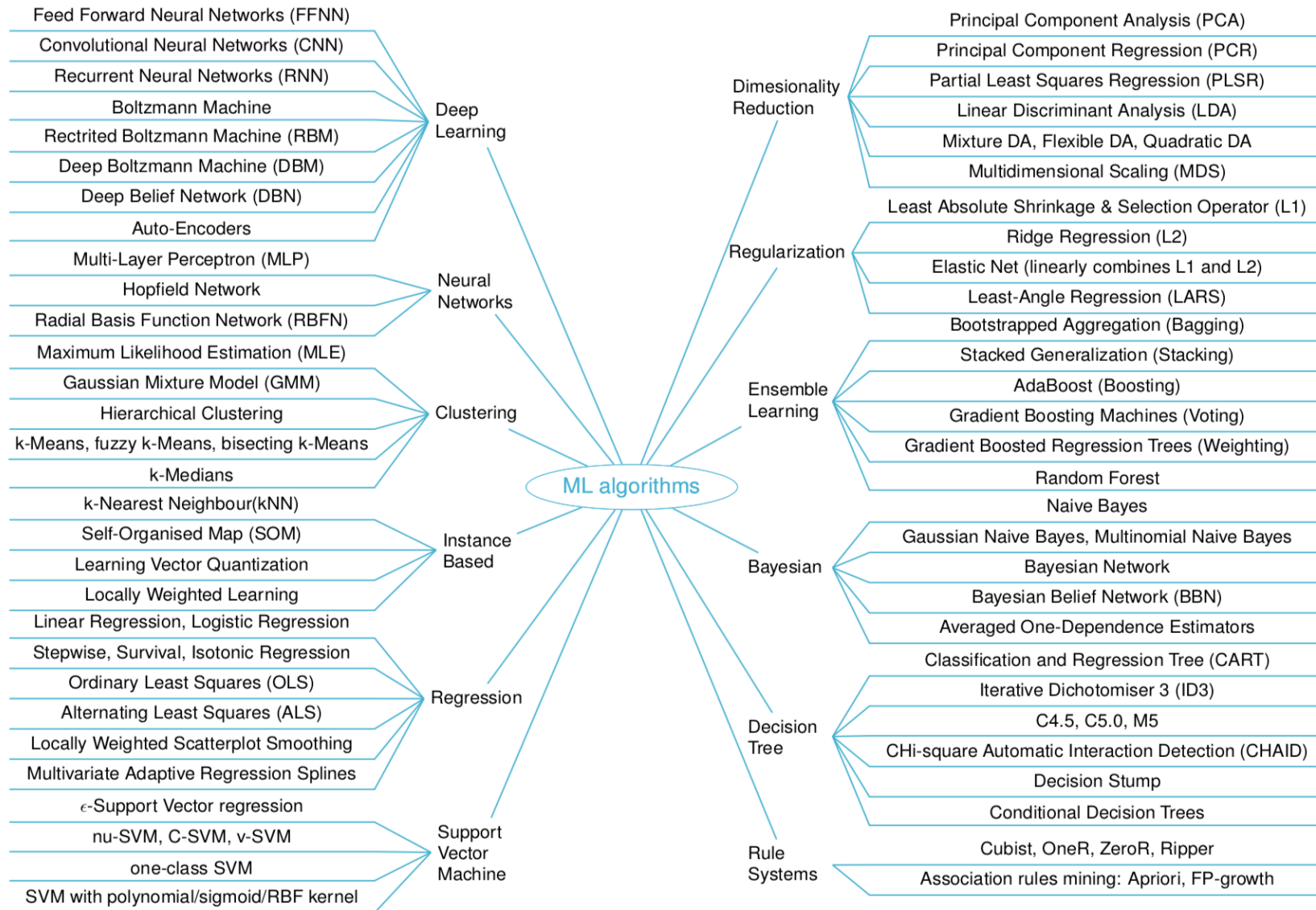


Fig. 3 Overview of Machine Learning algorithms

## 2.2. From Neural Networks to Deep Learning

As previously described, 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, with basically three layer categories: input layers, hidden (middle) layers and output layers. NNs can be divided into shallow (one hidden layer) and deep (several 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. It can then learn to relate these representations to output variables and therefore learn to predict. In fact, mathematically, NNs are capable of learning any mapping function (known as the universal approximation theorem [Cybenko 1989]).

### 2.2.1. Deep Neural Networks and Deep Learning architectures

Deep neural networks (DNNs) are considered to be capable of learning high-level features with more complexity and abstraction than shallower NNs due to their larger number of hidden layers. Defining a network architecture and training routine are two dependent problems that have to be focused in a problem solving with NNs in order to achieve high predictive accuracy [Goodfellow 2016] [Lisa 2015] [Schmidhuber 2015]. Defining network architectures involves setting certain fine-grained details like 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 routines for training involves into setting 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). Some common DL architectures are:

- **Feed Forward Neural Network (FFNN)**, also known as (deep) neural network (DNN) or multi-layer perceptron (MLP), is the most common type of NNs. FFNNs work well on tabular (i.e. transactional) data, which is the main data type in financial and insurance companies [H2O.ai 2017].
- **Convolutional Neural Network (CNN or ConvNet)** is traditionally a good choice for image data [Lazebnik 2017]. The most simple architecture consists on a stack on convolutional and pooling layers with a fully connected layer at the end.
- **Recurrent Neural Network (RNN)** is a kind of folded NN. RNNs are distinguished from FFNNs the fact information can also flow backwards through feedback loops. One of the most popular blocks for building layers of RNNs are Long Short Term Memory (LSTM) units, which are composed of a cell, an input gate, an output gate and a forget gate. Some also popular blocks like Gated Recurrent Units (GRU) are improvements over the LSTM block. RNNs can deal well with context-sensitive, sequential or time-series data.

- **Boltzmann Machine** is a kind of generative models with stochastic approach [Salakhutdinov 2009]. It is a network of symmetrically coupled stochastic binary units and its name is due to the use of Boltzmann distribution in statistics. Boltzmann Machine is a counterpart of Hopfield nets. **Restricted Boltzmann Machine** (RBM) interprets the NN as not a feedforward one, but a bipartite graph where the idea is to learn joint probability distribution of hidden and input variables. A RBM has no connections between hidden units. **Deep Boltzmann Machine** (DBM) comprises undirected Markov random fields with many densely connected layers of latent variables. DBMs have the potential of learning internal representations that become increasingly complex.
- **Deep Belief Network** (DBN) is a kind of directed sigmoid belief networks with many densely connected layers of latent variables. Belief network is probabilistic directed acyclic graphical model, which represents a set of variables and their conditional dependencies via a directed acyclic graph.
- **Autoencoder** is a kind of network useful for learning feature representations in an unsupervised manner. An autoencoder first compresses (encodes) the input vector to fit in a smaller representation, and then tries to reconstruct (decode) the input back.

More about DL architectures is available in [Veen 2016].

Year	Model	Number of layers	Top 5 error at ILSVRC (%)	Description
1990	LeNet	4	-	LeNet is one of the first commercial successful CNN applications [LeCun 1998]. It was deployed in ATMs to recognize digits for check deposits. The most well known version is the LeNet-5 (Fig. 4).
2012	AlexNet	8	16.4	AlexNet is the first CNN to win the ILSVRC and used GPUs to train the network.
2013	ZF Net	8	11.7	ZF Net won ILSVRC 2013 [Zeiler 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 2015]. The interesting of the architecture is the number of filters doubles after each maxpool layer. This reinforces the idea of shrinking spatial dimensions, but growing depth.
2015	GoogLeNet (Inception)	22	6.7	GoogLeNet (also referred to as the Inception) won the ILSVRC 2014 [Szegedy 2015]. It introduced an inception module composed of parallel connections witch drastically reduced the number of parameters. From this point, CNN architecture became more than only sequential stacks. Today, GoogleLeNet 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 ILSVRC 2015 [He 2016] [He 2016b], being the first network to surpass human-level accuracy with a top-5 error rate below 5%. It uses residual connections i.e. shortcut module or bypass to go even deeper. ResNets have been used as a starting point to further develop new architectures, like Wide ResNets [Zagoruyko 2016] or DenseNets [Huang 2017].
2016	SqueezeNet	14	14.0	SqueezeNet focuses in heavily reducing model size using deep compression [Iandola 2016] without losing accuracy. The result is a network with roughly the same classification accuracy as AlexNet but with 510 times less in the memory requirement (0.5 MB of memory tested on the ILSVRC 2012 dataset).

Table 1 Deep Learning timeline through the most well-known models

### 2.2.2. Deep Learning timeline through the most well-known models

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 Deep Learning (DL) successes of the 2010s are believed to be under the confluence of three main factors:

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

Many DNN models have been developed over the past two decades [Deshpande 2017] [Kalray 2017] [Sze 2017]. Each of these models has a different network architecture in terms of number of layers, layer types, layer shapes and connections between layers. In Table 1 we present a timeline of some iconic computer vision models over the past years. Some of them will be presented along with their performance in a well-known computer vision challenge, the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [Russakovsky 2015].

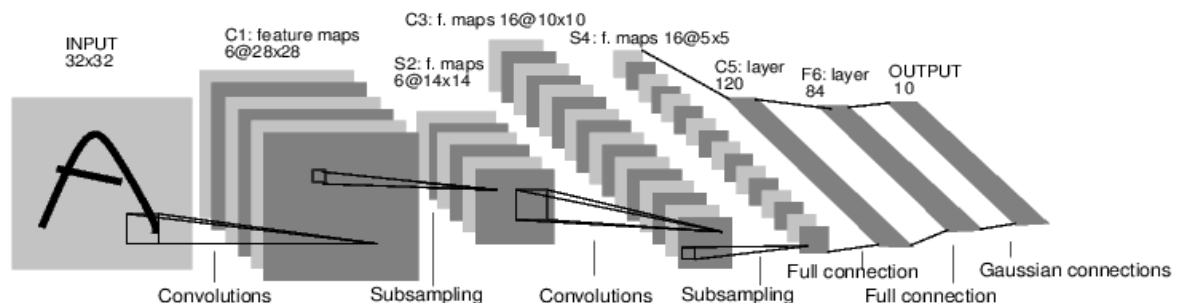


Fig. 4 The LeNet-5 model [LeCun 1998]

### 2.2.3. Problems in Deep Learning and advanced algorithmic solutions

As mentioned above, one factor that influences current DL success is the new algorithmic advances, which have alleviated some problems that prevented NN applications from properly working. Those problems are:

- **The vanishing gradient problem** describes the fact that the gradient signal barely reaches the first layers after being propagated from the final layers, causing very slow learning in very deep networks. This problem can be alleviated by using several components:
  - *Rectified linear units* (ReLU) as activation functions to improve gradient backward flow (in contrast with sigmoid and hyperbolic-tangent functions).
  - *Shortcut connections* to connect distant part of the networks through identity mappings.
  - *Batch normalization layers* to improve the internal covariance shift problem [Ioffe 2015]. These methods have enabled to train networks as deep as 1000 layers [He 2016b].



- **The overfitting problem** describes the fact that network perform very well on the training set but fail to generalize test data. This can be fought by using:
  - *Weight decay* (e.g. L1, L2), which penalizes layers weights that become too high.
  - *Dropout layers* that block a random number of units in a layer (usually around 50%) each training cycle [Srivastava 2014]. The random blocking provides incentive for kernels to learn more robust filters. At inference, all connections are used with corrective constant that is equal to the percentage of blocked connections.
  - *Network pruning* which represents a way to combat overfitting by discarding unimportant connections [Han 2016]. The advantage of this method is that the number of parameters is significantly reduced leading to smaller memory and energy requirements.
- **The model size problem** describes the fact that modern high performing models can be highly computationally and memory intensive. DNNs can have millions or even billions of parameters due to their rich connectivity. This increases the computational, memory bandwidth, and storage demands. To minimize these demands one can use:
  - *Deep compression* significantly reduces the network parameters with the aim of reducing memory requirements so the whole deep network model can fit into the on-chip memory. The process starts with network pruning when the importance of each connection is learnt. It is followed by quantizing the network and weight sharing and finally Huffman coding is applied [Han 2015].
  - *Sparse computation* that imposes the use of sparse representations along the network allow memory and computation benefits.
  - *Low precision data types* [Konsor 2012], smaller than 32-bits (e.g. half-precision or integer) with experimentation even with 1-bit computation [Courbariaux 2016]. This speeds up algebra calculation as well as greatly decreasing memory consumption at the cost of a slightly less accurate model. In these recent years, most DNNs are starting to support 16-bit and 8-bit computation.

## 2.3. Accelerated computing and Deep Learning

In addition to the above-mentioned algorithmic advances, the other two factors responsible of the DL successes are the availability of huge amounts of data and computing power. DL needs to use specialised hardware with low-latency interconnects in accelerated computing i.e. massive parallel architecture, extension of the Single Instruction Multiple Data (SIMD) paradigm with large scale multi-threading, streaming memory and dynamic scheduling. Better hardware would allow to scale training beyond current data and allow to create bigger and more accurate models.

The current mainstream solution [NVidiaAC] has been to use *Graphics Processing Unit* (GPU) as general purpose processors (GPGPU). GPUs provide a massive parallelism for large-scale DM problems, allowing scaling vertically algorithms to data volumes not computable by traditional






approaches [Cano 2017]. GPUs are effective solutions for real-world and real-time systems requiring very fast decision and learning, such as DL, especially image processing.

Beside that, the use of *Field Programmable Gate Array* (FPGA) [Lacey 2016] and the recently announced Google TPU2 (Tensor Processing Unit Second-Generation) for inference and training also constitute an interesting alternative [TPU2 2017]. Other IT companies also start 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].

Vertical scalability for large-scale data mining is still limited due to the GPU memory capacity, which is up to 16GB on the NVidia Pascal architecture at the moment. 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). Integration of MapReduce frameworks with GPU computing may overcome many of the performance limitations and it is open challenges and future research [Cano 2017].

### 2.3.1. Accelerated libraries

The main feature of the many-core accelerators such as GPU is their massively parallel architecture allowing them to speed up computations that involve matrix-based operations, which is a 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 (Table 2).

Library	Description
<b>CUDA</b> 	The <b>NVIDIA CUDA</b> (Compute Unified Device Architecture) [Cuda] 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] provides a development environment for creating high performance GPU-accelerated applications.
<b>cuDNN</b> 	The <b>NVIDIA CUDA Deep Neural Network library (cuDNN)</b> [cuDNN], 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 e.g. Caffe2, MatLab, CNTK, TensorFlow, Theano, and PyTorch.
<b>OpenCL</b> 	<b>OpenCL</b> (Open Computing Language) developed by Khronos provides compatibility across heterogeneous hardware from any vendor [OpenCL].
<b>Intel MKL</b>	<b>Intel MKL</b> (Intel Math Kernel Library) [MKL] optimizes code with minimal effort for future generations of Intel processors. It is compatible with many


Library	Description
	compilers, languages, operating systems, and linking and threading models. It accelerates math processing routines, increase application performance, and reduce development time. This ready-to-use math library includes: linear algebra, Fast Fourier Transforms (FFT), Deep Neural Networks (DNN), vector statistics and data fitting, vector math and miscellaneous solvers.

Table 2 Accelerated libraries from the biggest worldwide manufactures

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

- **OpenMP:** application programming interface (API) that supports multi-platform shared memory multiprocessing programming [OpenMP]. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behaviour.
- **Open MPI:** open source, freely available implementation of the MPI specifications [Open MPI]. 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.

An 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. More details about accelerators and accelerated computing will be available in Deliverable D4.1 Available Technologies for accelerators and HPC.

In the recent years the accelerators have been successfully used in many areas e.g. text, image, sound processing and recognition, life simulations as ML/NN/DL applications. Applicable areas of DNNs are:

- **Image and video processing:** satellites images (such as fires, droughts, crops diseases, urban development), space (telescope images), biology image recognition (such as plant, cells, bacteria), medical image recognition (such as magnetic resonance imaging, computer tomography, roentgen images, sonography), automatic picture or audio annotations [Hafiane 2017];
- **Speech and language:** text processing and recognition, speech recognition, machine translation, natural language processing;
- **Security:** biometrics authentication (such as people, faces, gait), anomaly detection, intrusion detection;
- **Business intelligence:** insurance, financial markets, stock and exchange rate predictions;
- **Robotics and videogames:** autonomous navigation (such as car, drone, plane, submarine), videogames (such as Atari, Dota, Starcraft).

### 2.3.2. Digital ecosystems and the embedding trend

Digital ecosystems consist of hardware, software, and services that create dependencies leading to user loyalty [Bajarin 2011]. In the context of DM using ML techniques in the Big Data era, the following ecosystems (Table 3) are frequently mentioned:





Ecosystem	Description
	<b>Python ecosystem</b> is built around Python programming language, which provides full features under the philosophy that the same libraries and code can be used for model development as well as in production. Python also has a complete scientific computing stack and a professional grade ML library for general purpose use.
	<b>Java ecosystems</b> is built around Java programming language, which has strong weight in business software development.
	<b>Hadoop/Spark ecosystem</b> is an ecosystem of Apache open source projects and a wide range of commercial tools and solutions that fundamentally change the way of Big Data storage, processing and analysis.
	<b>Cloud ecosystem</b> is a complex system of interdependent components that work together to enable cloud services. In a hybrid cloud environment, an organization combines services and data from a variety of models to create a unified, automated, and well-managed computing environment. Cloud ecosystems provide virtualisation solution e.g. docker i.e. installing all the DL frameworks takes time, so download a docker image is faster with the same running environment on different machines.

Table 3 Digital ecosystems

Furthermore, there are virtual environments at various levels e.g. in Python ecosystem as well as in Cloud ecosystems. The trend is to build an isolated environment for each prototyping stack in order to avoid interfering with other system configurations.

## 3. State-of-the-art of Machine Learning frameworks and libraries

The number of ML algorithms, as well as their different software implementation, is extensively high. Many software tools for DM using ML techniques have been in development for the past 25 years [Jovic 2014]. Their common goal is to facilitate the complicated data analysis process and to propose integrated environments on top of standard programming languages. Beside that, tools are designed for various purposes: as analytic platform, predictive systems, recommender systems, processors (from image, sound or language). A number of them are oriented to large-scale data, fast processing or streaming. Other ones are specialized for NNs and DL. There is no single tool suitable for every problem and often a combination of them is needed to solve it. Fig. 5 provides a comprehensive overview of ML frameworks and libraries.

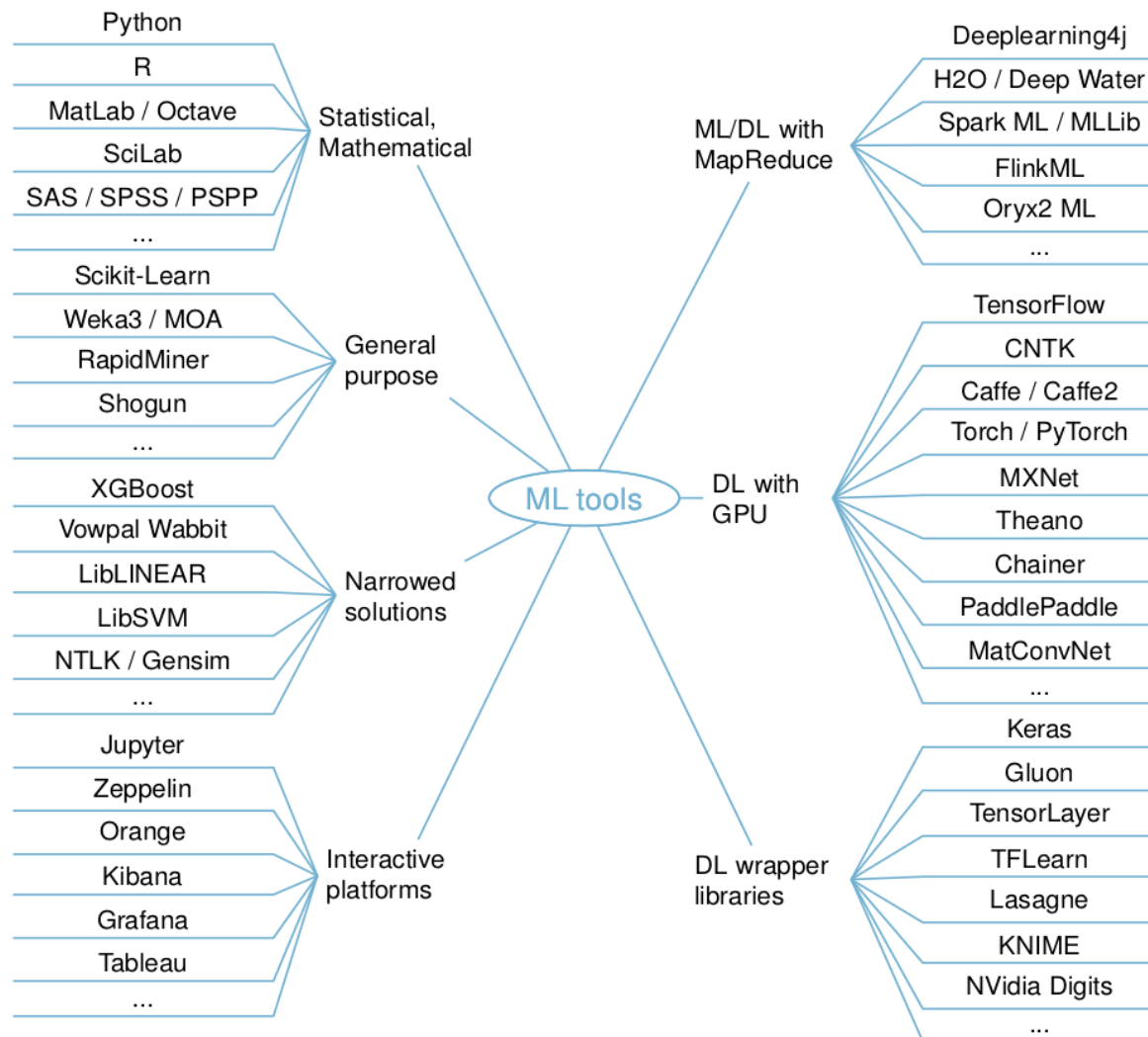


Fig. 5 Overview of Machine Learning frameworks and libraries

In following sections, 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 have dataflow architectures (pipeline or workflow). Some of them have graphical integrated environments (GUI), others prefer an API approach or both. The software development in ML/DL direction is highly dynamic with various abstraction layers of implementations as depicted in Fig. 6.

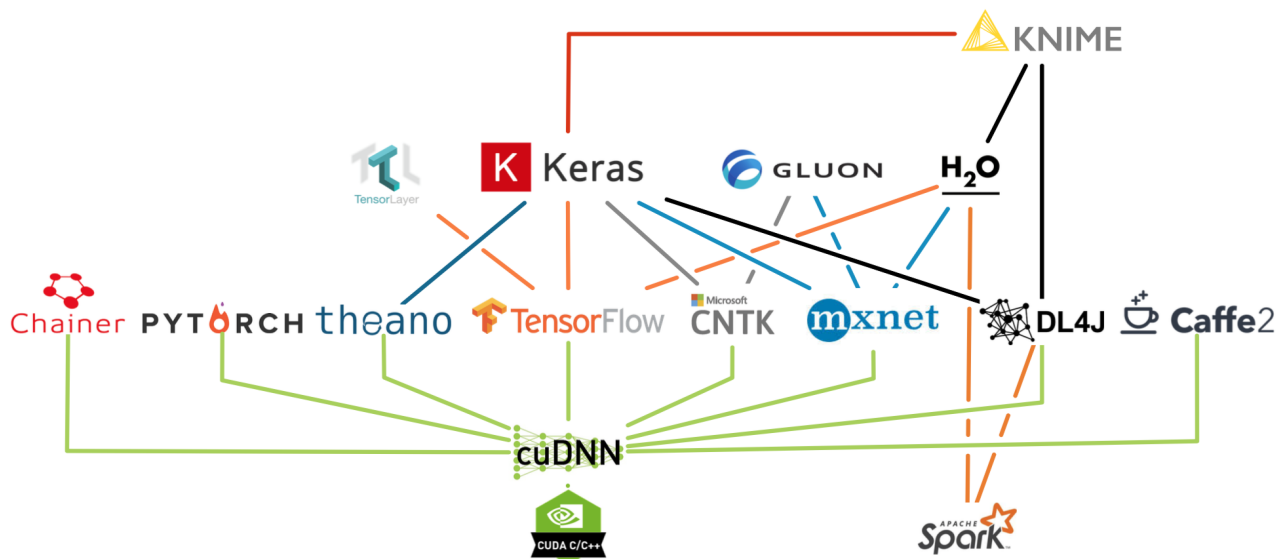













Fig. 6 Machine Learning and Deep Learning frameworks and libraries layering based on abstraction implementation levels

The details of these tools are presented in the part 3.2 (DL frameworks and libraries with GPU support) and the part 3.3 (ML/DL frameworks and libraries integrated with MapReduce). The following part 3.1 is concentrated into the state-of-the-art of ML/NN frameworks and libraries, which do not require special hardware or infrastructure supports. Nevertheless, these tools can utilise multi-CPU power to deal with large-scale data.

A short overview of Sections 3.1, 3.2 and 3.3 is provided in Table 4, Table 5 and Table 6 respectively. These tables summarise the frameworks and libraries capabilities so users can choose appropriate products for tackling their problems. Each tool is also described and evaluated separately afterwards in more detail.

### 3.1. General Machine Learning frameworks and libraries

The application of ML to diverse areas of computing is gaining popularity rapidly, because of the increasing availability of free and open source software enabling ML algorithms to be implemented easily. There is a wide range of open source ML frameworks, which enable to build, implement and maintain impactful research and development in many life areas.

Tool	Type	Creator	Licence	Platform	Written in	Interface	Algorithm coverage	Workflow	Usage	Popularity
<b>Shogun</b> 	ML library	G. Rätsc, S. Sonnenburg 	GNU GPLv3	UNIX Windows Mac OS	C++	Python, Octave, R, Java, Scala, Lua, C#, Ruby	High	API	Academic	Low
<b>RapidMiner</b> 	ML/NN/DL framework	R. Klinkenber, I. Mierswa, S. Fischer 	Proprietary	UNIX Windows	Java	Python, R, GUI, API	High	Yes	Academic Industrial	High
<b>Weka</b> 	ML/NN Framework		GNU GPLv3	Windows UNIX Mac OS	Java, GUI	Java, GUI, API	High	Yes	Academic	High
<b>Scikit-Learn</b> 	ML/NN Framework	D. Cournapeau  	BSD	UNIX Windows Mac OS	Python C++	Python, API	High	Yes API	Academic	High
<b>LibSVM</b>	SVM library Classification	C.C. Chang, C.J. Lin  國立臺灣大學 National Taiwan University	BSD 3-clause	UNIX Windows Mac OS GPU	C/C++	Java, Matlab, Octave, R, Python, C#, Perl, Ruby, Node.js, JavaScript, Lisp, CLisp, Haskell, PHP, Android	Low	No	Academic	Medium
<b>LibLinear</b>	ML library Classification	R.E.Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, C.J. Lin  國立臺灣大學 National Taiwan University	BSD 3-clause	UNIX Windows Mac OS	C/C++	Matlab, Octave, Java, Python, Ruby, Perl, R, Labview, Common Lisp, Scilab, CLI	Low  Linear SVM, Linear Regression	No	Academic	Medium
<b>Vowpal Wabbit</b>	ML library Fast	J. Langford	BSD 3-clause	UNIX Windows	C++ own MPI	API	Low	No	Academic Industrial	Medium



Tool	Type	Creator	Licence	Platform	Written in	Interface	Algorithm coverage	Workflow	Usage	Popularity
	out-of-core incremental			Mac OS Hadoop HPC	library for Hadoop AllReduce					
<b>XGBoost</b> <i>dmlc</i> <b>XGBoost</b>	ML library boosting ensemble	T. Chen 	Apache 2.0	UNIX Windows Mac OS Hadoop	C++	C++, Java, Python, R, Julia	Low	API	Academic Industrial	Medium

Table 4 Machine Learning and Neural Networks frameworks and libraries without special supports



### 3.1.1. Shogun



Shogun is the oldest open-source general purpose ML library that offers a wide range of efficient and unified ML methods [Shogun] [ShogunGoogle] [Sonnenburg 2010] built on an architecture written in C++. It is licensed under the terms of the GNU GPLv3 license. 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 (HMM), statistical testing, clustering, distance counting, FFNNs and model evaluations and many more. It has been under active development since 1999, with involving maintenance (the current version is 6.1.3, 12.2017). Original authors are Gunnar Rätsch from Max Planck Society for the Advancement of Science, and Sören Sonnenburg from Berlin Institute of Technology. Currently, Shogun is developed by a diverse team of volunteers and it is fiscally sponsored project of NumFOCUS since 2017. The main idea behind Shogun is that the underlying algorithms are transparent and accessible and anyone should be able to use for free. It was successfully used in speech and handwriting recognition, medical diagnosis, bioinformatics, computer vision, object recognition, stock market analysis, network security, intrusion detection, and many other. Shogun can be used transparently in many languages and environments 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 and examples.
- For academic use only.

### 3.1.2. RapidMiner



RapidMiner is a general purpose data science software platform for data preparation, ML, DL, text mining, and predictive analytics [Mierswa 2003] [Rapid]. 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). RapidMiner (formerly YALE, Yet Another Learning Environment) was developed starting in 2001 by Ralf



Klinkenberg, Ingo Mierswa, and Simon Fischer at the Artificial Intelligence Unit of the Technical University of Dortmund. It is developed on an open core model. It is written in the Java programming language and is a cross-platform framework. RapidMiner supports interactive mode (GUI), command-line interface (CLI) and Java API. RapidMiner is mainly proprietary commercial product since version 6.0. However it offers a free edition limited to one logical processor and 10,000 data rows, which is available under the AGPL license.

For large-scale data analytics, RapidMiner supports unsupervised learning in Hadoop [Radoop], supervised learning in memory with scoring on the cluster (SparkRM), and supervised learning and scoring with native algorithms on the cluster. In this case, the algorithm coverage is narrowed into Naive Bayes, iterative Naive Bayes, linear regression, logistic regression, SVM, decision tree, and random forest and clustering using k-means and fuzzy k-means.

#### Strong points

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

#### Weak points

- Proprietary product for large problem solutions.

### 3.1.3. Weka3



Weka collects a general purpose and very popular wide set of ML algorithms implemented in Java and engineered specifically for DM [Weka] . 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. While Weka isn't aimed specifically at Hadoop users and Big Data processing, it can be used with Hadoop thanks to a set of wrappers produced for the most recent versions of Weka3. At the moment, it still does not support Apache Spark, but only MapReduce. Clojure [Clojure] users can also leverage Weka, thanks to the Clj-ml library [Clj-ml]. Related to Weka, Massive Online Analysis (MOA) 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, well-maintained, involving wide set of algorithms with learning schemes, models and algorithms.
- It comes with GUI and 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 for Big Data, text mining, and semi-supervised learning.
- Weak for sequence modelling e.g. time-series.

### 3.1.4. Scikit-Learn



Scikit-Learn is widely known as a well-maintained, open source and popular Python ML tool, which contains comprehensive algorithm library included incremental learning [Scikit]. It extends the functionality of NumPy and SciPy packages with numerous DM algorithms. It also uses the Matplotlib package for plotting charts. 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. Since April 2016, Scikit-Learn is provided in jointly-developed Anaconda [Anaconda] for Cloudera project on Hadoop clusters [AnaCloudera]. 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. Scikit-Learn provides access to the following sorts of functionality: classification, regression, clustering, dimensionality reduction, model selection and preprocessing.

#### Strong points

- General purpose, open source, commercially usable, well-maintained and popular Python ML tools.
- Support from big IT companies (Google) and institutions (INRIA).
- 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

- Small datasets, API-oriented only, command-line interface requires Python programming skills.
- The library does not support GPU and has only basic tools for neural networks.

### 3.1.5. LibSVM



國立臺灣大學  
National Taiwan University

LibSVM is a specialized library for Support Vector Machines (SVM). Its development started in 2000 by Chih-Chung Chang and Chih-Jen Lin at National Taiwan University [Chang 2011] [LibSVM]. It is

written in C/C++ but has also Java source code. The learning tasks are 1) support vector classification (binary and multi-class), 2) support vector regression, and 3) distribution estimation. Supported problem formulations are: C-Support Vector Classification, v-Support Vector Classification, distribution estimation (one-class SVM),  $\epsilon$ -Support Vector Regression, and v-Support Vector Regression. 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 was successfully used in computer vision, NLP, neuro-imaging, and bioinformatics (since 2000 to 2010 with 250 000 downloads). It is also included in some DM environments: RapidMiner, PCP, and LIONsolver. The SVM learning code from the library is often reused in other open source ML toolkits, including GATE [Gate], KNIME [Knime], Orange [Orange] and scikit-learn. The library is very popular at open source ML community (released under the 3-clause BSD license). LibSVM version 3.22 released on December, 2016.

#### 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, well-maintained and specialised tool with high popularity in open source ML community.

#### Weak points

- LibSVM training algorithm does not scale up well for very large datasets in comparison with LibLinear or Vowpal Wabbit [Zygmunt 2014]. It takes  $O(n^3)$  time in the worst case and around  $O(n^2)$  on typical cases.
- Limited to problems, with which SVM deals well.

### 3.1.6. LibLinear



LibLinear is a library designed for solving large-scale linear classification problems. It was developed starting in 2007 by Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang and Chih-Jen Lin at National Taiwan University [Fan 2008] [LibLinear]. The library is written in C/C++. The supported ML tasks are logistic regression and linear SVM. The supported problem formulations 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 LR and also L2-SVM, LibLinear implements a trust region Newton method. For multi-class problems, LibLinear implements the one-vs-the-rest strategy and Crammer and Singer method. The SVM learning code from the library is often reused in other open source ML toolkits, including GATE, KNIME, Orange and scikit-learn. The library is very popular in the open source ML community (it is released under the 3-clause BSD license). LibLinear version 2.20 was released on December, 2017.

#### Strong points

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

#### Weak points

- Limited to logistic regression and linear SVM.

### 3.1.7. Vowpal Wabbit



Vowpal Wabbit (or VW) is efficient scalable implementation of online ML and for support of various incremental ML methods [VW] [VWAzure]. It is an open-source fast out-of-core learning system originally developed by John Langford at Yahoo! Research, and currently being developed at Microsoft Research. VW is one of the offered ML options in Microsoft Azure. It is notable for its many features including e.g., reduction functions, importance weighting, selection of different loss functions, optimization algorithms. VW has been used to learn a tera-feature (1012) data-set on 1000 nodes in one hour, 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 (the hashing trick).
- 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 of the strong industrial laboratory, compiled C++ code, well-maintained (github), well-supported.

#### Weak points

- The number of available ML methods is sufficient but limited.
- API-oriented environment only.

### 3.1.8. XGBoost



XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable [Chen 2016] [DMLC] [Mitchell 2017] [XGBoost]. It is an open-source software library that provides the gradient boosting framework 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. The XGBoost library 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 to strong by iteratively learning.

#### Strong points

- High execution speed and model performance.
- Parallelisation of tree construction using all of CPU cores during trainings.
- 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 only a boosting library that works for tabular data. Therefore it will not work for image recognition, NLP or computer vision.

### 3.1.9. Interactive data analytics and data visualisation

Tools in this category display analytics results in interactive way, so they can facilitate the understanding of difficult concepts and support decision makers for researchers and data scientists. There are many data visualization packages in various levels in R or Python e.g. **Matplotlib**, **Plotly**, **Seaborn**, **ggplot**, **Bokeh**, and so on.

In recent years, web-based notebooks/applications have been increasing in popularity. They are integrated with data analytic environments to create and share documents that contain data-driven live code, equations, visualisations and narrative text. The most well-known are Jupyter notebook (formerly iPython notebook) and Zeppelin.



**Jupyter** notebook [Jupyter] is the open-source application supporting e.g. creation and sharing documents ("notebooks"), code, source equations, visualisation and text descriptions for data transformation, numerical simulations, statistical modelling, data visualisation and ML.



**Zeppelin** is an interactive notebook designed for the processing, analysis and visualization of large data sets [Zeppelin], providing 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.

The next popular tools belong to open source data analytics, reporting and integration platforms such as Kibana, Grafana and Tableau.



**Kibana** is the data visualisation front end for the Elastic Stack, complementing the rest of the stack that includes Beats, Logstash and Elasticsearch [Kibana]. With the version 5.x release of the Elastic Stack, Kibana now includes Timelion for interactive time series charts.



**Grafana** is the chosen DevOps tool for many real time monitoring dashboards of time series metrics [Grafana]. It has powerful visualisations 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].

In conclusion, there are also rich options of interactive tools, which are designed for many different purposes.

### 3.1.10. Other tools including data analytic frameworks and libraries

The number of frameworks and libraries coupled with the analytical process using ML/NN/DL techniques is quite high. A relevant subset of them are described below.



**MatLab** (matrix laboratory) is a multi-paradigm numerical computing environment. It uses a proprietary programming language developed by MathWorks [MatLab]. 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] and SciLab [SciLab].



**SAS** (Statistical Analysis System) began as a project to analyse agricultural data at North Carolina State University in 1966 [SAS]. 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]. It was developed in 1968 and was acquired by IBM in 2009. An open source alternative of SPSS is GNU **PSPP** [PSPP].



**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]. R is ease of use and extensible via packages. The Comprehensive R Archive Network offers more than 10000 packages [R-CRAN].





**Python** is a programming language created by Guido van Rossum and first released in 1991 [Python]. Python is successfully used in thousands of real-world business applications around the world e.g. Google and YouTube. The primary 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.



**NumPy** is the fundamental package for scientific computing with Python [NumPy]. 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]. 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]. 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** is a leading platform for building Python programs to work with human language data [NLTK]. It comes with a suite of text processing libraries for classification, tokenisation, stemming, tagging, parsing, and semantic reasoning.

## 3.2. Deep Learning frameworks and libraries with GPU support

Many popular ML frameworks and libraries already offer the possibility to use GPU accelerators to speed up learning process with supported interfaces such as TensorFlow, CNTK, Theano, Keras, Caffe, Torch, DL4J, MXNet, Chainer and many more [DLwiki] [Felice 2017] [Kalogeiton 2017]. Some of them also allow to use optimised libraries such as CUDA (cuDNN), and OpenCL to improve the performance even further. The main feature of the many-core accelerators is a massively parallel architecture allowing them to speed up computations that involve matrix-based operations. The GPGPU interest can be found in many other life large-scale simulation packages with dynamic progress developments.

Tool	Type	Creator	Licence	Mobile solution	Accelerated libraries	Backends	Written in	Interface	Computational graph	Usage	Popularity
TensorFlow 	Numerical framework		Apache 2.0	Yes (TensorFlow Lite)	CUDA OpenMP		C++ Python	Python, C++*, Java*, Go*  <i>*not fully covered</i>	Static with small support for dynamic graph (TensorFlow Fold)	Scientific Industrial	Very High (growing fast)
Keras 	Library	F. Chollet  	MIT	No	as backend	TensorFlow Theano CNTK DL4J MXNet	Python	Python	Static	Scientific Industrial	High (growing very fast)
CNTK 	Framework Toolkit		Open source Microsoft permissive license	Limited	CUDA Open MPI MKL		C++	Python, C++, BrainScript ONNX	Static	Scientific Industrial	Medium (growing fast)
Caffe 	Framework	Y. Jia 	BSD 2-clause	No	CUDA		C++	C++, Python, MatLab	Static	Scientific Industrial	High (growing fast)
Caffe2 	Framework	Y. Jia 	Apache-2.0	Yes	CUDA		C++	C++, Python, ONNX	Static	Mobile computing	Medium-Low (growing fast)
Torch 	Framework	R. Collobert, K. Kavukcuoglu, C. Farabet	BSD	No	CUDA OpenMP OpenCL		C++ Lua	C, C++, Lua, LuaJIT, OpenCL	Static	Scientific Industrial	Medium-Low (stagnating)
PyTorch 	Library	A. Paszke, S. Gross, S. Chintala, G. Chanan	BSD	No	CUDA		Python C	Python ONNX	Dynamic	Scientific Industrial	Medium (growing very fast)
MXNet 	Framework		Apache-2.0	No	CUDA OpenMP		C++	C++, Python, Julia, Matlab, JavaScript,	Dynamic dependency scheduler	Scientific Industrial	Medium (growing fast)







Tool	Type	Creator	Licence	Mobile solution	Accelerated libraries	Backends	Written in	Interface	Computational graph	Usage	Popularity
								Go, R, Scala, Perl, ONNX			
Theano 	Numerical framework	Y. Bengio 	BSD	No	CUDA OpenMP		Python	Python	Static	Scientific Industrial	Medium-Low (stagnating)
Chainer 	Framework		Open source, Owner's permissive license	No	CUDA MKL-DNN opt. for Intel architecture		Python	Python	Dynamic	Scientific Industrial	Low (stagnating)

Table 5 Deep Learning frameworks and libraries with GPU support

The popularity and trend measures are the subject to change according to the high dynamic development of DL framework and tools. The estimation of these values was based on Github repository stargazing [Jolav 2018].

### 3.2.1. TensorFlow



TensorFlow is an open source software library for numerical computation using data flow graphs [TensorFlow]. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. TensorFlow was created and is maintained by 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 programming interfaces includes Python and C++ with plans for Java, GO, R, and Haskell APIs. It is also supported in Google and Amazon cloud environment. TensorFlow is designed for large-scale distributed training and inference. 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++. Unlike other DL libraries that are mainly focused on research (such as Theano) 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 TensorFlow lightweight solution for mobile and embedded devices [TensorflowLite]. It enables on-device ML inference with low latency and a small binary size but has coverage for a limited set of operators. It also supports hardware acceleration with the Android Neural Networks API.

#### Strong points

- *By far the most popular DL tool*, open source, fast involving, well-supported by the strong industrial company (Google).
- Powerful numerical library for dataflow programming that provides the basis for DL research and development.
- Efficiently works with mathematical expressions involving multi-dimensional arrays.
- Very well documented.
- GPU/CPU computing, mobile computing, high scalability of computation across machines and huge data sets.
- Higher layer of abstraction than Theano.

#### 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 tries to alleviate this problem), and lacks symbolic loops.

### 3.2.2. Keras



Keras is a minimalist Python library for DL that can run on top of TensorFlow, CNTK, Theano, beta version with MXNet and announced DeepLearning4j [Keras]. It was developed with a focus on enabling fast experimentation and is released under the MIT license. Keras runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. Keras is developed and maintained by Francois Chollet, a Google engineer using four guiding principles:

1. *User friendliness and minimalism.* Keras is an API designed for human beings with user experience front and center. 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, easy to debug, and allows easy extensibility.

#### Strong points

- Open source, fast involving, well-supported by strong industrial companies.
- Allows to quickly define DL models; Keras may become the standard API for DL, it has very good documentation.
- Clean and convenient way to create a range of 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 not 100% working.
- Less projects available online than Caffe.

### 3.2.3. CNTK



Microsoft Cognitive Toolkit (CNTK) is commercial-grade distributed DL with large-scale datasets from Microsoft Research [CNTK]. 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, it is running on both 64-bit Linux and Windows operating systems using Python, C#, C++ and BrainScript API. CNTK implements stochastic gradient descent (SGD) learning with automatic differentiation and parallelization across multiple GPUs and servers.

#### Strong points

- Open source, fast evolving, well-supported by a strong industrial company (Microsoft).
- Higher performance in comparison with Theano and Tensorflow when running on multiple machines.
- Supports the Open Neural Network Exchange (ONNX) format, which will allow easy moving between CNTK, Caffe2, PyTorch, MXNet and other DL tools. ONNX is co-developed by Microsoft and Facebook.

#### Weak points

- Limited capability on mobile devices.

### 3.2.4. Caffe



Caffe is a DL framework made with expression, speed, and modularity in mind. It is developed by Yangqing Jia at the Berkeley Artificial Intelligence Research (BAIR) and by community contributors [Caffe]. 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), memory, file system, 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 FFNN and excellent implementation of CNN for image processing.
- Fastest DL library on CPU, GPU out-of-the-box training.
- A number of pre-trained networks directly from the Caffe Model Zoo, available for immediate use.
- Easy to code (API/CLI) with Python and MatLab interface.
- Well-acceptable from research community.

#### Weak points

- It is not good RNN i.e. for text, sound and time-series data.
- Cumbersome for complicated DNN models i.e. GoogleLeNet and ResNet.

- Custom layers must be written in C++.

### 3.2.5. Caffe2



**Caffe2**

Caffe2 is a lightweight, modular, and scalable DL framework developed by Yangqing Jia and his team at Facebook [Caffe2]. 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 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 all claim to support Caffe2 due to its robust scalable character in production.
- Supports the Open Neural Network Exchange (ONNX) format, which will allow easy moving between CNTK, Caffe2, PyTorch, MXNet and other DL tools.

#### Weak points

- Harder for DL beginners in comparison with PyTorch [Caffe2vsPyTorch].
- Without dynamic graph computation.
- Limited in flexibility.

### 3.2.6. Torch



Torch is a scientific computing framework with wide support for ML algorithms based on the Lua programming language [Torch]. It has been under active development since 2002. The original authors are Ronan Collobert, Koray Kavukcuoglu, Clement Farabet [Collobert 2002]. Torch has been developed using an object-oriented paradigm and implemented in C++. Nowadays, its API is also written in Lua language (Lua is a multi-paradigm scripting language created in 1993 by R. Lersalimschy, L. de Figueiredo, and W. Celes at the University of Rio de Janeiro). Lua language is used as a wrapper for optimized C/C++ and CUDA code. Its core is made up by tensor library which provides both CPU and GPU backends. Current version Torch7, 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 2011]. The framework supports parallelism on multi-core CPUs via OpenMP, and on GPUs via CUDA. It is aimed on large-scale learning (speech, image, and video applications), and affords supervised learning, unsupervised learning, reinforced learning, NNs, optimization, graphical models, image processing. Torch is supported and used by Facebook, Google, DeepMind, Twitter, and many other organizations. The framework is freely available under a BSD license.

#### 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.

### 3.2.7. PyTorch



PyTorch is a Python library for GPU-accelerated DL [PyTorch]. The library is a Python interface of the same optimized C libraries that Torch uses. It has been developed by Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan (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 (it provides Tensors that can run either on the CPU or the GPU, highly accelerating compute), and DNNs built on a tape-based autograd system. It has become popular by allowing certain complex architectures to be built easily [Deeplearning4j, 2018]. 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], and Chainer [Chainer]. The library was used by both the scientific and the industrial community. An engineering team at Uber has built Pyro, a universal probabilistic programming language using PyTorch as its back end. A DL training site fast.ai announced the future switching to be based on PyTorch rather than Keras-TensorFlow [Patel, 2017]. The library is freely available under a BSD license and it 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 [Caffe2vsPyTorch].
- Supports the Open Neural Network Exchange (ONNX) format, which will allow easy moving between CNTK, Caffe2, PyTorch, MXNet and other DL tools.

#### Weak points

- Still without mobile solution in comparison with Caffe2.

### 3.2.8. MXNet

Apache MXNet is a DL framework designed for both efficiency and flexibility [MXNet]. It allows mixing symbolic and imperative programming to maximize efficiency and productivity. MXNet is open source library for DL with broad API language support for R, Python, Julia and other languages [Chen 2015]. 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]. 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 is licensed under an Apache-2.0 license. MXNet is supported by major public cloud providers. It also supports an efficient deployment of a trained model to low-end devices for inference, such as mobile devices (using Amalgamation Amalgamation), IoT devices (using AWS Greengrass), Serverless (Using AWS Lambda) or containers.

#### Strong points

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

#### Weak points

- APIs are not very user-friendly in some cases. However, it has two user-friendly wrappers (Keras and Gluon).
- Excellent for parallel production work, but less flexible for DL research.



### 3.2.9. Theano

**theano** Theano is a pioneering DL tool (development started in 2007) supporting GPU computation. It is an open source project released under the BSD license [Theano]. It is actively maintained (although no longer developed) by the LISA group (now MILA Montreal Institute for Learning Algorithms [MILA]) 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, well-maintained 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.
- Lack for mobile platform and other programming API's.
- The active development at the current level has ended after the 1.0.0 version on November 2017 as announced by Y. Bengio [MILA 2017]. The Theano maintenance would continue (the current version is 1.0.1 on January 2018).

### 3.2.10. Chainer



Chainer is a python-based DL framework aiming at flexibility [Chainer] [Tokui 2015]. It provides automatic differentiation APIs based on the define-by-run approach i.e. dynamic computational graphs as well as object-oriented high-level APIs to build and train NNs. The difference from other famous DL framework like Tensorflow or Caffe is that Chainer constructs NN dynamically. It also supports CUDA/cuDNN using CuPy (motivation NumPy + CUDA = CuPy) for high performance training and inference. Chainer core team of developers work at Preferred Networks, Inc. a ML startup with engineers mainly from the University of Tokyo. Chainer supports CNN, RNN DL architectures. DL frameworks are usually built based on the “*Define-and-Run*” scheme i.e. at the beginning a computational graph is constructed and a network is statically defined and fixed. Chainer’s design is based on the principle “*Define-by-Run*” i.e. network is not predefined at the beginning, but is dynamically defined on-the-fly (i.e. dynamic computational graph or DCG). Chainer contains libraries for industrial applications e.g. ChainerCV (library for DL in Computer Vision), ChainerRL (deep reinforcement learning library built on top of Chainer), ChainerMN (scalable multi-node distributed DL with Chainer where linear speed-up up to 128 GPUs), etc. Intel Chainer with MKL-DNN backend is approx. 8.35 times faster than NumPy backend (according to Chainer benchmarks).



#### Strong points

- Dynamic computational graph (Define-by-Run).
- Provides libraries for industrial applications.
- Strong investors such as Toyota, FANUC, NTT, etc.

#### Weak points

- No support for higher order gradients.
- DCG is generated every time also for fixed networks, still no optimization even for static part of graphs.

### 3.2.11. Wrapper frameworks and libraries

As mentioned above, Keras is a wrapper library for DL libraries of lower level of implementation abstractions. There are also more wrapper libraries, some of them are quite popular, other have unique design. These wrapper libraries differ each from others in transparency levels towards underlying frameworks or libraries as well as they are chosen based on users preferences and popularity.



Tensorflow has a lot of wrappers. External wrapper packages are **TensorLayer** [TensorLayer], **TFLearn** [TFLean] and **Keras**. Wrappers from Google are **Sonnet** (Deepmind) [Sonnet] and **PrettyTensor** [PrettyTensor]. Wrappers are **TF-Slim** [TF-Slim], **tf.contrib.learn**, **tf.layers**, and **tf.keras** [TensorFlow].



**Gluon** is a wrapper for MXNet [Gluon]. 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 from Amazon Web Services (AWS) and Microsoft's AI. It is released under Apache 2.0 licence.



**NVidia Digits** - Deep Learning GPU Training System [DIGITS] is 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 plug-ins. DIGITS 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 from the results browser for deployment. DIGITS is mainly interactive (GUI). It provides availability of pre-trained models such as AlexNet, GoogLeNet, LeNet and UNET from the DIGITS Model Store and is released under BSD 3-clause license.



**Lasagne** is lightweight library to build and train NNs in Theano with six principles: Simplicity, Transparency, Modularity, Pragmatism, Restraint and Focus [Lasagne]. Other wrappers for Theano are **Blocks** and **Pylearn2**. Due to the fact that Theano is not under active development, the popularity of these wrappers is bound to decrease.

### 3.2.12. Other DL frameworks and libraries with GPU supports

**PaddlePaddle** (PARallel Distributed Deep LEarning) is an open source, efficient, flexible and scalable DL platform, which is originally developed by Baidu scientists and engineers for the purpose of applying DL to many Baidu products [PaddlePaddle]. At Baidu, PaddlePaddle has been deployed into products and services with a vast number of users, including ad click-through rate (CTR) prediction, large-scale image classification, optical character recognition (OCR), search ranking, computer virus detection, recommendation, etc. It supports a wide range of NN architectures and optimization algorithms. It is easy to configure complex models such as neural machine translation model with attention mechanism or complex memory connection.

**MatConvNet** is a MatLab toolbox implementing CNNs for computer vision applications [MatConvNet] developed by the Oxford computer vision team and other research institutions. It is simple and integrating MatLab GPU support, and can run and learn CNNs with similar results to top scores in the ImageNet challenge. Many pre-trained CNNs models e.g. VGG, AlexNet for image classification, segmentation, face recognition, and text detection are available. An important feature of MatConvNet is making available the CNN building blocks as easy-to-use MatLab commands. MatConvNet does not support for RBMs and DBNs and it does not have OpenMP and OpenCL supports. The next weak point of the product is the proprietary character of MatLab.

**The NVIDIA Deep Learning SDK**, which is a part of the NVIDIA toolkit, provides powerful tools and libraries for designing and deploying GPU-accelerated DL applications. It includes libraries for DL primitives, inference, video analytics, linear algebra, sparse matrices, and multi-GPU communications. The NVIDIA CUDA Deep Neural Network (cuDNN) library is a GPU-accelerated library of primitives for deep neural networks, It accelerates widely used DL frameworks, including Caffe2, MATLAB, CNTK, TensorFlow, Theano, PyTorch, etc. The next tools in the NVIDIA Deep Learning SDK are Deep Learning Inference Engine (TensorRT), Deep Learning for Video Analytics (DeepStream SDK), Linear Algebra (cuBLAS), Sparse Matrix Operations (cuSPARSE) and Multi-GPU Communication (NCCL).

The development of DL frameworks and libraries is quite high dynamic with many interesting involving products. The popularity/trend movement of DL frameworks and libraries at the end of 2017 are depicted in Fig. 7 borrowed from <https://towardsdatascience.com>. It is difficult to make forecast in this fast changing ecosystem but we can see to main trends emerging in the use of DL frameworks: 1) a trend backed by Google with uses Keras for fast prototyping and Tensorflow for production, and 2) a trend backed by Facebook which uses Pytorch for prototyping and Caffe2 for production.

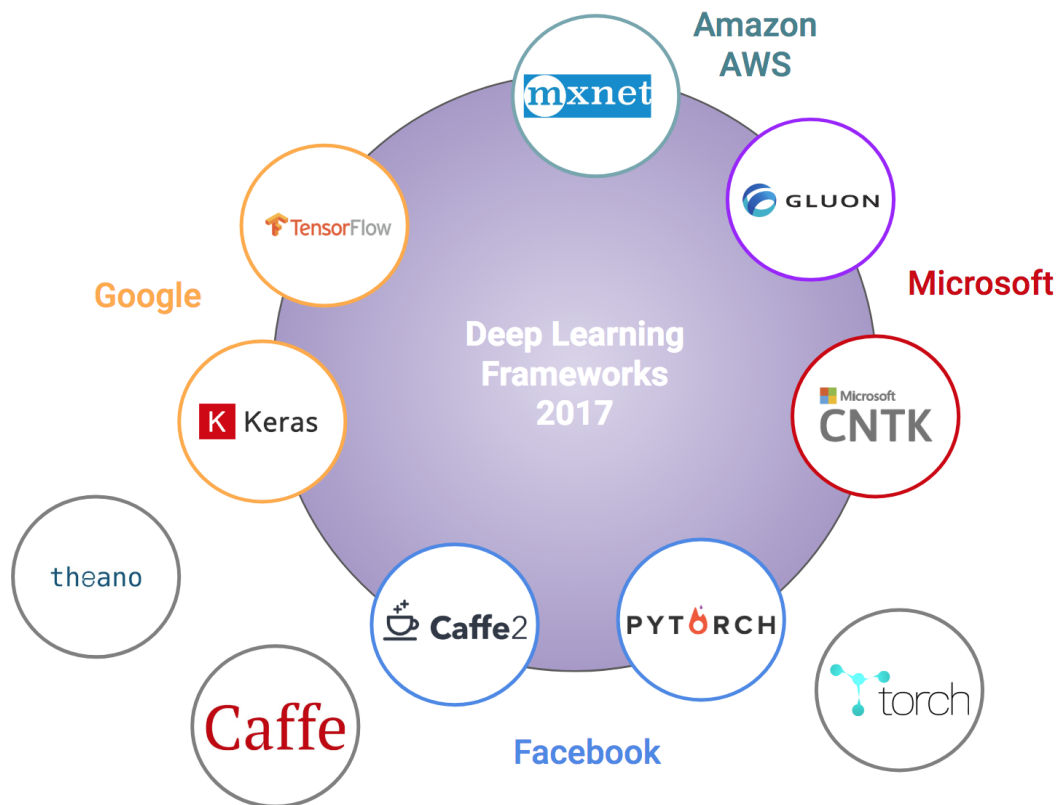


Fig. 7 State of open source Deep Learning frameworks at the end of 2017 [Bakker 2017]

The extensive number of deep learning frameworks makes it challenging to develop tools in one framework and use them in other frameworks (frame 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 DL frameworks Caffe2, PyTorch, MXNet and Microsoft CNTK (Fig. 8).



Fig. 8 ONNX Open ecosystem for interchangeable AI models

### 3.3. Machine Learning and Deep Learning frameworks and libraries with MapReduce

Recently, newly distributed frameworks have emerged to address the scalability of algorithms to 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 are 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 and GPUs are not mutually exclusive technologies, although they aim at different scaling purposes [Cano 2017]. These technologies can complement each other and target complementary computing scopes such as ML and DL [Skymind 2017], however here is still a lot of limitations and challenges.





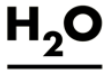



Tool	Type	Creator	Licence	Written in	Backends	Interfaces	Algorithm coverage	Usage	Popularity
<b>DL4J</b> 	DL library for Java		Open source Apache 2.0	Java, Scala  CUDA, cuDNN support via JNI	Integration with Spark	Java, Scala, Clojure, Python	Medium (DL)	Industrial	Medium
<b>Apache Spark MLlib &amp; ML</b> 	ML/NN library		Open source Apache 2.0	Scala	Integration with Python (NumPy), R	Java, Scala, Python, R	Medium (ML)	Industrial	Low
<b>H2O</b> 	General purpose framework for ML/DL and Big Data analytics		Open source Apache 2.0	Java GUI API	TensorFlow MXNet Caffe	REST API JSON+HTTP Java, Scala , Python, R	Medium (ML/DL)	Industrial	High
<b>KNIME</b> 	Analytic platform		GNU GPLv3	Java with CUDA support	Integration with R, Python, Weka, Keras, H2O, DL4J	GUI wrapper	as backends	Academic Industrial	Low

Table 6 Machine Learning and Deep Learning frameworks and libraries integrated with MapReduce

### 3.3.1. Deeplearning4j



**DL4J**

Deeplearning4j or DL4J is distinguished from other ML/DL frameworks and libraries. 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 YARN, Hadoop's distributed, application management framework [DL4J] [Skymind 2017]. The library consists of several sub-projects for developers 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), tool 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 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 format similarly as it is in Hadoop via InputFormat.

#### Strong points

- The distinguished advantage of DL4j is it uses the whole power 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 platform, where the Java ecosystem is predominate in business software development.
- Rich set of DL architectures CNN, RNN (RNTN, LTSM), RBM and DBN i.e, excellent capabilities for image recognition, fraud detection and NLP.

#### Weak points

- Java/Scala are not the most popular language in the DL/ML community like Python.
- Currently, it gains less overall interest than H2O in Big Data and Spark community.

### 3.3.2. Apache Spark MLlib and ML



Firstly, Apache introduced **Mahout** built on the top of MapReduce. Mahout was mature and came with many ML algorithms. However, ML algorithms generally use many iterations making Mahout run very slowly. Apache, then, introduced **Spark MLlib** and **Spark ML** built on top of Spark ecosystem on Hadoop, which making them much

faster than Mahout. Spark MLlib contains old RDD-based API (Resilient Distributed Dataset). RDD is the Spark 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 DataFrame can be specified as SQL queries, which is convenient for developers with SQL background. Moreover, Spark SQL provides Spark more information about the structure of both the data and the computation being performed than Spark RDD API. Spark ML brings a 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. Spark MLlib is slowly being deprecated in the maintenance mode and most likely will be removed in a future major release.

Spark MLlib and Spark ML contain ML 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. They also contain tools for linear algebra, statistics and data handling. Except 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 to learn offline and then apply the model on streaming data online.

#### 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) aslo 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 and Spark ML are quite young ML libraries in involving state. They are not very popular and the number of ML algorithm implementation is not very high.



### 3.3.3. H2O, Sparkling and Deep Water

**H<sub>2</sub>O** H2O, Sparkling Water and Deep Water are developed by H2O.ai (formerly 0xdata) [H2O]; they are Hadoop compatible frameworks for 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 Map/Reduce 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 instance. Hadoop maven 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.

**H2O:** Except the REST API and 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 (similar to Jupyter). Data source for the framework are natively local FS, Remote File, HDFS, S3, JDBC, others through generic HDFS API. Although the ML algorithm coverage is not high, they are optimised to run over Big Data and cover the need of the target companies i.e. banks and insurance sectors. In details, H2O is used by 8/10 top banks for pattern-based Anti-Money Laundering (AML), fraudulent behaviour detection, real-time personalised product recommendation; 7/10 top insurance companies for risk group and claim classification automation, customer churn reduction, customer retention analysis, insurance fraud alert system and usage-based insurance telematics; and 4/10 top healthcare companies for real-time preventive care, cancer detection or personalised medicine development.

Regarding the 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** (see Fig. 9) 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.

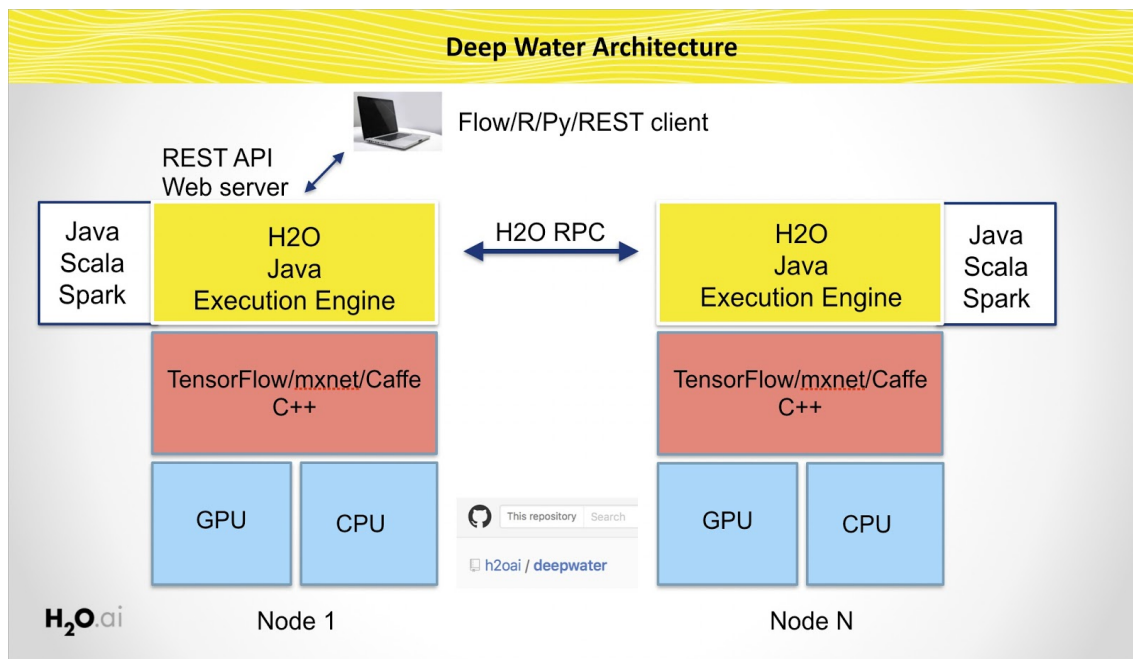


Fig. 9 H2O Deep Water architecture [H2Odeepwater]

#### 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 wider 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 user interface for H2O, do not support direct interaction with Spark.
- H2O is more general purpose and aims at different scalable DM in comparison with (specific) DL libraries e.g. TensorFlow or DL4j.

### 3.3.4. Other frameworks and libraries coupled with MapReduce



**FlinkML** is a part of Apache Flink, which is an open-source framework for distributed stream and batch data processing [Flink]. FlinkML aims to provide a set of scalable ML algorithms and an intuitive 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 its ability to process data streams in real time. The main difference between Spark and Flink lies in the way each framework 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 Lambda architecture built on Apache Spark and Apache Kafka for real-time large scale ML [Oryx2]; 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 (ALS, random decision forests, k-means).



**KNIME** (Konstanz Information Miner) is the data analytic, reporting and integration platform of the Knime AG, Switzerland [KNIME]. It integrates various components for ML and DM through its modular data pipelining concept through GUI allowing assembly of nodes for data preprocessing (ETL - Extraction, Transformation and Load), for 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 also allows for wrappers calling other code in addition to providing nodes that allow to run Java, Python, Perl and other programming languages; and integration with Weka, R, Python, Keras (DL), H2O (ML/DL), DL4J (DL, Hadoop/Spark). It has considerable community supports i.e. it is used by over 3000 organizations in more than 60 countries.

## 4. Conclusions

Machine Learning (ML), especially its subfield Deep Learning (DL), had many amazing advances in the recent years, and may lead to technological breakthroughs that will be used by billions of people. The software development in this field is fast changing with a great number of open-source software from academic, industry, start-up, and open source communities.

As a new computing model, DL with GPU support is changing how software is developed and how it runs. Nowadays, ML algorithms learn from huge amount of real-world examples in variety formats. DL is about designing and training NNs. After a computationally consuming training, NNs can be deployed in data centers to infer, predict and classify from new incoming data presented to them. Trained NNs can also be deployed into intelligent IoT devices to understand the world. The deployment of trained NNs require smaller computational resources in comparison with the development phase. The new computing model in the Big Data era requires massive data processing and massive parallelism supports that are capable to scale computation effectively and efficiently according to the real need.

ML and DM are research areas of computer science with fast involving development due to the advances in data analysis research in the Big Data era. When the number of ML algorithms is extensive and growing, the number of their realizations through ML/NN/DL frameworks and libraries are extensive and growing too. The short outcome of the document is follows.

- Most of the DL/NN framework development is done at the world's largest software companies such as Google, Facebook, and Microsoft. These companies dispose a huge data, high performance infrastructure, human intelligence and investment resources. Their most popular DL tools are TensorFlow, Microsoft CNTK, Caffe/Caffe2, Torch/PyTorch, and MXNet. Apart from them, other DL tools such as Chainer, Theano, DL4J, and H2O from other companies and research institutions, are also interesting, well-supported and suitable for industrial use.
- There are many high level wrapper libraries built on a top of above-mentioned DL tools (e.g. Keras, TensorLayer, Gluon) suitable for convenient DL development.
- Big Data ecosystems such as Apache Spark/Flink and Cloudera Oryx 2 contain build-in ML libraries for large-scale data mining (mainly for tabular data). These ML libraries are currently in involving state but the power of the whole ecosystem is significant.
- Every tool (including traditional general purpose ML tools) provides a way to process large-scale data.
- As of the year 2018, the Python is the most popular programming language for ML/DL applications. It is used as general purpose language for research, development and production, at small and large scales. The majority of scientific data analytic and ML/DL tools are either Python tools or support Python interfaces.
- The trend shows also a high number of interactive data analytics and data visualisation tools supporting decision makers.

It is needed to notice that using these kind of tools is not the only way to build compute-intensive applications or to do data analytics and data mining. Self made code packages can do the same job. The price for this way is, of course, the time and the efforts spent on the code development and code maintenance process.

There is a challenge of managing multiple tools, multiple approaches from divergent ML/DL user communities in different applicable areas. The challenge is hard because of exposing a unified, comprehensive, efficient and coherent platform, that is capable to scale computation dynamically and on-demands. The combined impact of new computing resources and techniques with an increasing avalanche of large datasets, is transforming many research areas. This evolution has many different faces, components and contexts, and our project DEEP Hybrid DataCloud will combine some of them to propose a new e-infrastructure framework able to address relevant challenges in research.

## 5. References

- [Bajarin 2011] Bajarin, B.: Why It's All About the Digital Ecosystem <https://techpinions.com/why-its-all-about-the-ecosystem/4567>
- [Bakker 2017] Bakker I.: Battle of the Deep Learning frameworks—Part I: 2017, even more frameworks and interfaces, Dec 2017, <https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750>
- [Bowlee 2017] Bowlee J.: Deep learning with Python (MachineLearningMastery.com), 2017
- [Cano 2017] Cano, A., 2017. A survey on graphic processing unit computing for large scale data mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.
- [Chang 2011] Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2(3), 27.
- [Chen 2015] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. and Zhang, Z., 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274.
- [Collobert 2002] Collobert, R., Bengio, S. and Mariéthoz, J., 2002. Torch: a modular machine learning software library (No. EPFL-REPORT-82802). Idiap.
- [Collobert 2011] Collobert, R., Kavukcuoglu, K. and Farabet, C., 2011. Torch7: A matlab-like environment for machine learning. In BigLearn, NIPS Workshop (No. EPFL-CONF-192376).
- [Courbariaux 2016] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R. and Bengio, Y., 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830.
- [CRIPS-DM 1999] CRIPS-DM Cross-Industry Standard Process for Data Mining EU FP4-ESPRIT 4, ID 24959, 1997-1999, [http://cordis.europa.eu/project/rcn/37679\\_en.html](http://cordis.europa.eu/project/rcn/37679_en.html)
- [Cybenko 1989] Cybenko, G. (1989) "Approximations by superpositions of sigmoidal functions", Mathematics of Control, Signals, and Systems, 2 (4), 303-314
- [DL4j 2018] Comparing Top Deep learning Frameworks: Deeplearning4j, PyTorch, TensorFlow, Caffe, Keras, MxNet, Gluon & CNTK, accessed Feb 2018, <https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch>
- [Deshpande 2017] Deshpande A.: Understanding CNNs Part 3 <https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>
- [Fan 2008] Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. "LIBLINEAR: A library for large linear classification." Journal of machine learning research 9, no. Aug (2008): 1871-1874.

- [Felice 2017] Mitch De Felice, Which Deep learning network is best for you, May 2017, <https://www.cio.com/article/3193689/artificial-intelligence/which-deep-learning-network-is-best-for-you.html>
- [Goodfellow 2016] Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. MIT press.
- [H2O.ai 2017] Deep learning (Neural Networks), 12.2017, <http://h2o-release.s3.amazonaws.com/h2o/rel-wheeler/2/docs-website/h2o-docs/data-science/deep-learning.html>
- [Hafiane 2017] Hafiane, A., Vieyres, P. and Delbos, A., 2017. Deep learning with spatiotemporal consistency for nerve segmentation in ultrasound images. arXiv preprint arXiv:1706.05870.
- [Han 2015] Han, S., Mao, H. and Dally, W.J., 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.
- [Han 2016] Han, S., Pool, J., Tran, J. and Dally, W., 2015. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems (pp. 1135-1143).
- [He 2016] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- [He 2016b] He, K., Zhang, X., Ren, S., & Sun, J. (2016, October). Identity mappings in deep residual networks. In European Conference on Computer Vision (pp. 630-645). Springer, Cham
- [Huang 2017] Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017, July). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (Vol. 1, No. 2, p. 3).
- [Iandola 2016] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. and Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. arXiv preprint arXiv:1602.07360.
- [Ioffe 2015] Ioffe, S. and Szegedy, C., 2015, June. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning (pp. 448-456).
- [Jolav 2018] Jolav, Github Star History - <https://codetabs.com/github-stars/github-star-history.html>
- [Jovic 2014] Jovic, A., Brkic, K. and Bogunovic, N., 2014, May. An overview of free software tools for general data mining. In Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on (pp. 1112-1117). IEEE.
- [Kalogeiton 2017] Kalogeiton, V., Lathuilière, S., Luc, P., Lucas, T., Shmelkov, K., Deep learning frameworks: TensorFlow, Theano, Keras, Torch and Caffe, January 2017, <https://project.inria.fr/deeplearning/files/2016/05/DLFrameworks.pdf>
- [Kalray 2017] Kalray: Deep learning for high-performance applications <http://www.eenewseurope.com/Learning-center/kalray-deep-learning-high->



- performance-applications, March, 2017
- [Konsor 2012] Konsor P.: Intel software | Developer zone | Performance Benefits of Half Precision Floats, Intel Software Development Zone, August, 2012 <https://software.intel.com/en-us/articles/performance-benefits-of-half-precision-floats>
- [Krizhevsky 2012] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
- [Lazebnik 2017] Lazebnik, L.: Convolutional Neural Network Architectures: from LeNet to ResNet, 2017, [http://web.engr.illinois.edu/~slazebni/spring17/lec01\\_cnn\\_architectures.pdf](http://web.engr.illinois.edu/~slazebni/spring17/lec01_cnn_architectures.pdf)
- [Lacey 2016] Lacey G., Taylor G. W., & Areibi, S. (2016). Deep learning on FPGAs: Past, Present, and Future. arXiv preprint arXiv:1602.04283.
- [LeCun 1998] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.
- [Lisa 2015] Deep Learning Tutorial, Release 0.1, D.L., September 2015. LISA lab. University of Montreal.
- [Liu 2016] Liu, J., Li, J., Li, W. and Wu, J., 2016. Rethinking Big data: A review on the data quality and usage issues. ISPRS Journal of Photogrammetry and Remote Sensing, 115, pp.134-142.
- [Mierswa 2003] Mierswa, I., Klinkenberg, R., Fischer, S. and Ritthoff, O., August 2003. A flexible platform for knowledge discovery experiments: Yale–yet another learning environment. In Proc. of LLWA (Vol. 2003, p. 2).
- [Mierswa 2017] Mierswa, I: What is Artificial Intelligence, Machine Learning, and Deep Learning, 2017, <https://rapidminer.com/artificial-intelligence-machine-learning-deep-learning/>
- [MILA 2017] MILA and the future of Theano, accessed Feb 2018, <https://groups.google.com/forum/#!msg/theano-users/7Poq8BZutbY/rNCIfvAEAwAJ>
- [Mitchell 2017] Mitchell R., Gradient Boosting, Decision Trees and XGBoost with CUDA, September 2017, <https://devblogs.nvidia.com/parallelforall/gradient-boosting-decision-trees-xgboost-cuda/>
- [Nasyrov 2017] Nasyrov D., June 2017, Deep Neural Networks. Theory. Convolutional Networks. <https://medium.com/pharos-production/deep-neural-networks-theory-convolutional-networks-332c28ab82ad>
- [Patel, 2017] Mo Patel: "When two trends fuse: PyTorch and recommender systems". O'Reilly Media, accessed Feb 2018, <https://www.oreilly.com/ideas/when-two-trends-fuse-pytorch-and-recommender-systems>
- [Piatetsky 2017] Piatetsky, G., Python vs R - <https://www.kdnuggets.com/2017/09/python-vs-r-data-science-machine-learning.html>
- [Russakovsky 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C., 2015. Imagenet large scale visual recognition challenge. International Journal of Computer



- Vision, 115(3), pp.211-252.
- [Salakhutdinov 2009] Salakhutdinov, R. and Hinton, G., 2009, April. Deep boltzmann machines. In Artificial Intelligence and Statistics (pp. 448-455).
- [Simonyan 2015] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
- [Schaul 2010] Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstieß, T. and Schmidhuber, J., 2010. PyBrain. Journal of Machine Learning Research, 11(Feb), pp.743-746.
- [Schmidhuber 2015] Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural networks, 61, pp.85-117.
- [Skymind 2017] Comparing Top Deep Learning Frameworks: Deeplearning4j, PyTorch, TensorFlow, Caffe, Keras, MxNet, Gluon and CNTK, November 2017, <https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch>
- [Sonnenburg 2010] Sonnenburg, S.Č., Henschel, S., Widmer, C., Behr, J., Zien, A., Bona, F.D., Binder, A., Gehl, C. and Franc, V.: "The SHOGUN machine learning toolbox." Journal of Machine Learning Research 11, no. Jun (2010): 1799-1802.
- [Srivastava 2014] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research, 15(1), pp.1929-1958.
- [Sze 2017] Sze, V., Chen, Y.H., Yang, T.J. and Emer, J., 2017. Efficient processing of deep neural networks: A tutorial and survey. arXiv preprint arXiv:1703.09039.
- [Szegedy 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
- [Tokui 2015] Tokui, S., Oono, K., Hido, S. and Clayton, J., 2015, December. Chainer: a next-generation open source framework for deep learning. In Proceedings of workshop on machine learning systems (LearningSys) in the 29th annual conference on neural information processing systems (NIPS) (Vol. 5).
- [TPU2 2017] First In-Depth Look at Google's New Second-Generation TPU, May 2017, <https://www.nextplatform.com/2017/05/17/first-depth-look-google-new-second-generation-tpu/>
- [Upfront 2015] The Upfront Analytics Team, May 2015, Data Mining Vs Artificial Intelligence Vs Machine Learning, <http://upfrontanalytics.com/data-mining-vs-artificial-intelligence-vs-machine-learning/>
- [Varangaonkar 2017] Varangaonkar A.: Top 10 Deep Learning Frameworks, May 2017. <https://datahub.packtpub.com/deep-learning/top-10-deep-learning-frameworks>
- [Veen 2016] Veen F. V.: The neural network Zoo, Sep 2016, <http://www.asimovinstitute.org/neural-network-zoo/>

- [Zagoruyko 2016] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.
- [Zeiler 2014] Zeiler, M.D. and Fergus, R., 2014, September. Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.
- [Zygmunt 2014] Zygmunt Z., Vowpal Wabbit, Liblinear/SBM and StreamSVM compared, 2014, <http://fastml.com/vowpal-wabbit-liblinear-sbm-and-streamsvm-compared/>

## 5.1. Links

- [autograd] autograd - Automatic differentiation | Efficiently computes derivatives of NumPy code, accessed Feb 2018, <https://github.com/HIPS/autograd>
- [AnaCloudera] Anaconda for Cloudera - Data Science with Python Made Easy for Big data, accessed Feb 2018, <http://know.continuum.io/anaconda-for-cloudera.html>
- [Anaconda] Anaconda | The Most Popular Python Data Science Platform, accessed Feb 2018, <https://www.anaconda.com/what-is-anaconda/>
- [Caffe] Caffe | Deep learning framework by Berkeley Artificial Intelligence Research (BAIR), accessed Feb 2018, <http://caffe.berkeleyvision.org/>
- [Caffe2] Caffe2 | A New Lightweight, Modular, and Scalable Deep Learning Framework, accessed Feb 2018, <https://caffe2.ai/>
- [Caffe2PyTorch] Caffe2 vs. PyTorch, Apr. 2017, <https://discuss.pytorch.org/t/caffe2-vs-pytorch/2022/5>
- [Chainer] Chainer | A Powerful, Flexible, and Intuitive Framework for Neural Networks, accessed Feb 2018, <https://chainer.org/index.html>
- [Clj-ml] Clj-ml | A machine learning library for Clojure built on top of Weka and friends, accessed Feb 2018, <https://github.com/antoniogarrote/clj-ml>
- [Clojure] The Clojure Programming Language, accessed Feb 2018, <https://clojure.org/>
- [CNTK] Microsoft Cognitive Toolkit (CNTK), an open source deep-learning toolkit, accessed Feb 2018, <https://docs.microsoft.com/en-us/cognitive-toolkit/>
- [Cuda] CUDA Zone | NVIDIA Development, accessed Feb 2018, <https://developer.nvidia.com/cuda-zone>
- [cuBLAS] Linear Algebra, accessed Feb 2018, <https://developer.nvidia.com/cublas>
- [cuDNN] NVIDIA cuDNN | GPU Accelerated Deep Learning, accessed Feb 2018, <https://developer.nvidia.com/cudnn>

[CudaToolkit]	NVIDIA CUDA Toolkit, accessed Feb 2018, <a href="https://developer.nvidia.com/cuda-toolkit">https://developer.nvidia.com/cuda-toolkit</a>
[cuSPARSE]	Sparse Matrix Operations, accessed Feb 2018, <a href="https://developer.nvidia.com/cuspars">https://developer.nvidia.com/cuspars</a>
[DeepStreamSDK]	Deep Learning for Video Analytics, accessed Feb 2018, <a href="https://developer.nvidia.com/deepstream-sdk">https://developer.nvidia.com/deepstream-sdk</a>
[DIGITS]	The NVIDIA Deep Learning GPU Training System, accessed Feb 2018, <a href="https://developer.nvidia.com/digits">https://developer.nvidia.com/digits</a>
[DL4J]	Deeplearning4j   The first commercial-grade, open-source, distributed deep-learning library written for Java and Scala, integrated with Hadoop and Spark, accessed Feb 2018, <a href="https://deeplearning4j.org/">https://deeplearning4j.org/</a>
[DLwiki]	Comparison of deep learning software, accessed Feb 2018, <a href="https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software">https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software</a>
[DMLC]	DMLC for Scalable and Reliable Machine Learning, accessed Feb 2018, <a href="http://dmlc.ml/">http://dmlc.ml/</a>
[Flink]	Apache Flink: Scalable Stream and Batch Data Processing, accessed Feb 2018, <a href="https://flink.apache.org/">https://flink.apache.org/</a>
[Gate]	GATE General Architecture for Text Engineering, accessed Feb 2018, <a href="https://gate.ac.uk/">https://gate.ac.uk/</a>
[Grafana]	Grafana - The open platform for analytics and monitoring, accessed Feb 2018, <a href="https://grafana.com/">https://grafana.com/</a>
[Gluon]	A clear, concise, simple yet powerful and efficient API for deep learning, accessed Feb 2018, <a href="https://github.com/gluon-api/gluon-api">https://github.com/gluon-api/gluon-api</a>
[Jupyter]	Project Jupyter, accessed Feb 2018, <a href="https://jupyter.org/">https://jupyter.org/</a>
[Keras]	Keras   High-level neural networks API, accessed Feb 2018, <a href="https://keras.io/">https://keras.io/</a>
[Kibana]	Kibana: Explore, Visualize, Discover Data   Elastic, accessed Feb 2018, <a href="https://www.elastic.co/products/kibana">https://www.elastic.co/products/kibana</a>
[KNIME]	KNIME - Open for Innovation, accessed Feb 2018, <a href="https://www.knime.com/">https://www.knime.com/</a>
[H2O]	0xdata - H2O.ai   Fast Scalable Machine Learning, accessed Feb 2018, <a href="http://h2o.ai/">http://h2o.ai/</a>
[H2Odeepwater]	Deep Water, accessed Feb 2018,

<https://github.com/h2oai/deepwater/blob/master/README.md>

- [Lasagne] Lightweight library to build and train neural networks in Theano, accessed Feb 2018, <https://github.com/Lasagne/Lasagne>
- [LibLinear] LIBLINEAR - A Library for Large Linear Classification, accessed Feb 2018, <https://www.csie.ntu.edu.tw/~cjlin/liblinear/>
- [LibSVM] LIBSVM - A Library for Support Vector Machines, accessed Feb 2018, <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- [NCCL] Multi-GPU Communication, accessed Feb 2018, <https://developer.nvidia.com/nccl>
- [NLTK] Natural Language Toolkit, accessed Feb 2018, <http://www.nltk.org/>
- [NumPy] NumPy | The fundamental package for scientific computing with Python, accessed Feb 2018, <http://www.numpy.org/>
- [NVidiaAC] NVIDIA Accelerated Computing, accessed Feb 2018, <https://developer.nvidia.com/computeworks>
- [NVidiaDLS] NVIDIA Deep Learning SDK, accessed Feb 2018, <https://developer.nvidia.com/deep-learning-software>
- [MatConvNet] MatConvNet | CNNs for MATLAB, accessed Feb 2018, <http://www.vlfeat.org/matconvnet/>
- [MatLab] MatLab | The Language of Technical Computing, accessed Feb 2018, <http://www.mathworks.com/products/matlab/>
- [MILA] Montreal Institute for Learning Algorithms, accessed Feb 2018, <http://mila.umontreal.ca/>
- [MLK] Intel MKL | Intel Math Kernel Library, accessed Feb 2018, <https://software.intel.com/en-us/intel-mkl/>
- [MXNet] Apache MXNet - A flexible and efficient library for deep learning, accessed Feb 2018, <https://mxnet.apache.org/>
- [Octave] GNU Octave Scientific Programming Language, accessed Feb 2018, <https://www.gnu.org/software/octave/>
- [ONNX] Open Neural Network Exchange format, accessed Feb 2018, <https://onnx.ai/>
- [Orange] Orange | Open source machine learning and data visualization for novice and expert. Interactive data analysis workflows with a large toolbox, , accessed

Feb 2018, <https://orange.biolab.si/>

[OpenCL]	OpenCL   Open Computing Language - The Khronos Group Inc., 2018, <a href="https://www.khronos.org/opencv/">https://www.khronos.org/opencv/</a> ; <a href="https://developer.nvidia.com/opencv">https://developer.nvidia.com/opencv</a>
[OpenMP]	OpenMP   API specification for parallel programming, accessed Feb 2018, <a href="http://www.openmp.org">www.openmp.org</a>
[Open MPI]	Open MPI   Open Source High Performance Computing, accessed Feb 2018, <a href="https://www.open-mpi.org/">https://www.open-mpi.org/</a>
[Oryx2]	Oryx2   Framework for real-time large scale machine learning, accessed Feb 2018, <a href="http://oryx.io/">http://oryx.io/</a>
[PaddlePaddle]	PaddlePaddle   PARallel Distributed Deep LEarning, accessed Feb 2018, <a href="http://www.paddlepaddle.org/">http://www.paddlepaddle.org/</a>
[Pandas]	Pandas   Python Data Analysis Library, accessed Feb 2018, <a href="https://pandas.pydata.org/">https://pandas.pydata.org/</a>
[PrettyTensor]	Pretty Tensor   Fluent Networks in TensorFlow, accessed Feb 2018, <a href="https://github.com/google/prettytensor">https://github.com/google/prettytensor</a>
[PyBrain]	PyBrain official web page, accessed Feb 2018, <a href="http://www.pybrain.org">http://www.pybrain.org</a>
[Python]	Python Programming Language, accessed Feb 2018, <a href="https://www.python.org/">https://www.python.org/</a>
[PyTorch]	PyTorch   Deep learning framework that puts Python first, accessed Feb 2018, <a href="http://pytorch.org/">http://pytorch.org/</a>
[Radoop]	Advanced Radoop Processes - RapidMiner Documentation, accessed Feb 2018, <a href="https://docs.rapidminer.com/radoop/overview/radoop-advanced.html">https://docs.rapidminer.com/radoop/overview/radoop-advanced.html</a>
[Rapid]	RapidMiner Open Source Predictive Analytics Platform, accessed Feb 2018, <a href="https://rapidminer.com/">https://rapidminer.com/</a>
[R-CRAN]	Comprehensive R Archive Network (CRAN), accessed Feb 2018, <a href="https://cran.r-project.org/">https://cran.r-project.org/</a>
[Rproject]	R Project for Statistical Computing, accessed Feb 2018, <a href="http://www.r-project.org/">http://www.r-project.org/</a>
[PSPP]	GNU PSPP for statistical analysis of sampled data, accessed Feb 2018, <a href="https://www.gnu.org/software/pspp/">https://www.gnu.org/software/pspp/</a>
[SAS]	SAS (previously Statistical Analysis System), accessed Feb 2018, <a href="https://www.sas.com/en_us/">https://www.sas.com/en_us/</a>

[Scikit]	Scikit-Learn Machine Learning in Python, accessed Feb 2018, <a href="http://scikit-learn.org/stable/">http://scikit-learn.org/stable/</a>
[SciLab]	SciLab - Open source software for numerical computation, accessed Feb 2018, <a href="https://www.scilab.org/">https://www.scilab.org/</a>
[SciPy]	SciPy   Python-based ecosystem of open-source software for mathematics, science, and engineering, accessed Feb 2018, <a href="https://www.scipy.org/">https://www.scipy.org/</a>
[Shogun]	Shogun official web page, accessed Feb 2018, <a href="http://www.shogun.ml/">http://www.shogun.ml/</a>
[ShogunGoogle]	Shogun Machine Learning Toolbox - Google Summer of Code Archive, accessed Feb 2018, <a href="https://summerofcode.withgoogle.com/archive/2017/organizations/4704476053110784/">https://summerofcode.withgoogle.com/archive/2017/organizations/4704476053110784/</a>
[Sonet]	DeepMind   Sonnet TensorFlow-based neural network library, accessed Feb 2018, <a href="https://github.com/deepmind/sonnet">https://github.com/deepmind/sonnet</a>
[SPSS]	SPSS, accessed Feb 2018, <a href="http://www.ibm.com/software/analytics/spss/">http://www.ibm.com/software/analytics/spss/</a>
[Tableau]	Tableau Software: Business Intelligence and Analytics, accessed Feb 2018, <a href="https://www.tableau.com/">https://www.tableau.com/</a>
[TensorFlow]	TensorFlow   An open-source software library for Machine Intelligence, accessed Feb 2018, <a href="https://www.tensorflow.org/">https://www.tensorflow.org/</a>
[TensorFlowLite]	TensorFlow Lite, accessed Feb 2018, <a href="https://www.tensorflow.org/mobile/">https://www.tensorflow.org/mobile/</a>
[TensorLayer]	TensorLayer   Deep Learning (DL) and Reinforcement Learning (RL) library extended from Google TensorFlow, accessed Feb 2018, <a href="https://tensorlayer.readthedocs.io/en/latest/#">https://tensorlayer.readthedocs.io/en/latest/#</a>
[TF-Slim]	TF-Slim   TensorFlow-Slim   Lightweight library for defining, training and evaluating complex models in TensorFlow, accessed Feb 2018, <a href="https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim">https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim</a>
[TFLean]	TFLearn   TensorFlow Deep Learning Library, accessed Feb 2018, <a href="http://tflearn.org/">http://tflearn.org/</a>
[TensorRT]	Deep Learning Inference Engine, accessed Feb 2018, <a href="https://developer.nvidia.com/tensorrt">https://developer.nvidia.com/tensorrt</a>
[Theano]	Theano <a href="http://deeplearning.net/software/theano/">http://deeplearning.net/software/theano/</a>
[Torch]	Torch   scientific computing framework for LUAJIT <a href="http://torch.ch/">http://torch.ch/</a>
[Weka]	Weka3: Data Mining Software in Java, accessed Feb 2018,

<http://www.cs.waikato.ac.nz/ml/weka/>

- [VW] Vowpal Wabbit open source fast learning system, accessed Feb 2018, [https://github.com/JohnLangford/vowpal\\_wabbit/wiki](https://github.com/JohnLangford/vowpal_wabbit/wiki)
- [VWAzure] Text Analytics and Vowpal Wabbit in Azure Machine Learning Studio, accessed Feb 2018, <https://azure.microsoft.com/en-in/documentation/videos/text-analytics-and-vowpal-wabbit-in-azure-ml-studio/>
- [Zeppelin] Apache Zeppelin, accessed Feb 2018, <https://zeppelin.apache.org/>



## 6. Glossary

### 6.1. List of Figures

- Fig. 1 CRISP-DM Cross-Industry Standard Process for Data Mining
- Fig. 2 Relations between Artificial Intelligence, Machine Learning, Neural Networks and Deep Learning
- Fig. 3 Overview of Machine Learning algorithms
- Fig. 4 The LeNet-5 model
- Fig. 5 Overview of Machine Learning frameworks and libraries
- Fig. 6 Machine Learning and Deep Learning frameworks and libraries layering based on abstraction implementation levels
- Fig. 7 State of open source DL frameworks at the end of 2017
- Fig. 8 ONNX open ecosystem for interchangeable AI models
- Fig. 9 H2O Deep Water architecture

### 6.2. List of Tables

- Table 1 Deep Learning timeline through the most well-known models
- Table 2 Accelerated libraries from the biggest worldwide manufactures
- Table 3 Digital ecosystems
- Table 4 Machine Learning and Neural Networks frameworks and libraries without special supports
- Table 5 Deep Learning frameworks and libraries with GPU support
- Table 6 Machine Learning and Deep Learning frameworks and libraries integrated with MapReduce

## 6.3. Acronyms

AI	Artificial Intelligence
ALS	Alternating Least Squares
API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
AWS	Amazon Web Services
CART	Classification And Regression Tree
CLI	Command-Line Interface
CNN	Convolutional Neural Networks
CNTK	(Microsoft) Cognitive Toolkit
CONV	CNN convolutional layers
CRIPS-DM	Cross-Industry Standard Process for Data Mining
cuDNN	CUDA Deep Neural Network
DBN	Deep Belief Network
DL	Deep Learning
DM	Data Mining
DNN	Deep Neural Networks
DCG	Dynamic Computational Graph
EC2	Elastic Compute Cloud
EOSC	European Open Science Cloud
FC	(CNN) fully connected layers
FFNN	Feed Forward Neural Networks
FGPA	Field-Programmable Gate Array
GMM	Gaussian Mixture Model
GNU GPL	GNU General Public License
GPU	Graphics Processing Unit
GPGPU	General Purpose Graphics Processing Unit
GUI	Graphical User Interface
KDD	Knowledge Discovery and Data mining
HDFS	Hadoop Distributed File System
HDF5	Hierarchical Data Format
HMM	Hidden Markov Model
ILSVRC	ImageNet Large-Scale Visual Recognition challenge (ImageNet challenge)
MKL	(Intel) Math Kernel Library
ML	Machine Learning
MLP	Multi-Layer Perceptron
NB	Naive Bayes
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
NN	Neural Network
ONNX	Open Neural Network Exchange
OS	Operating System

RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RDD	Resilient Distributed Dataset
RMSprop	Root Mean Square Propagation
RNN	Recurrent Neural Networks
SCG	Static Computational Graph
SGD	Stochastic Gradient Descent
SIMD	Single Instruction Multiple Data
SVM	Support Vector Machines
TF-IDF	Term Frequency–Inverse Document Frequency
TPU	(Google) Tensor Processing Unit
VM	Virtual Machine
YARN	(Apache Hadoop) Yet Another Resource Negotiator