**BigDL**

<https://github.com/intel-analytics/BigDL>

<https://bigdl-project.github.io/master/#whitepaper/>

"BigDL is a distributed deep learning library for Apache Spark; with BigDL, users can write their deep learning applications as standard Spark programs, which can directly run on top of existing Spark or Hadoop clusters.

Spark doesn't support DL natively. Bigdl is a framework. Analtyics zoo is the platform to build models.

BigDL, a new deep learning framework with a focus on Apache Spark, only works on Intel chips."

**Caffe**

<https://github.com/BVLC/caffe>

<https://www.bizety.com/2018/10/29/tensorflow-vs-caffe/>

"Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR)/The Berkeley Vision and Learning Center (BVLC) and community contributors.

Caffe sets itself apart from other deep learning frameworks through its modularity and the fact that it is designed for scale. It has been described as an “un-framework” due to its flexibility and modularity.

Caffe2 was released in 2017 by Facebook as the successor to the original Caffe. However, Caffe continues to be available as a separate framework.

Caffe2 improves Caffe 1.0 in several ways:

Improved support for large-scale distributed training

mobile deployment

new hardware support (as well as CPU and CUDA)

flexibility for future directions, including quantized computation

stress tested by the huge scale of Facebook applications

The first Caffe framework was especially useful for large-scale product use cases due to its unparalleled performance and well tested C++ codebase. Caffe has various design choices inherited from its initial use case: conventional CNN applications. As new computation patterns have emerged, however, in particular, distributed computation, mobile, reduced precision computation, and some non-vision use cases, its design has shown several limitations.

With Caffee, the MPI library must be used. It was initially built for breaking apart applications on massive multi-node supercomputers, which can mean that implementing an MPI version, which is running a Caffe training process is not always straightforward.

Caffe does not have a straightforward deployment model for installation. It has to be developed from source.

Caffe does not offer a “high-level API for building models” says Schumacher, something that TensorFlow does offer. This can make Caffe harder to experiment with,

As Caffe’s GPU support currently offers no tools for Python, all training must be performed via a C++ based command line interface.

The original Caffe framework remains very popular for its community, its computer vision developments and its Model Zoo, essentially a selection of pre-trained models.

Caffe is not intended for other deep-learning applications such as text, sound or time series data."

**Chainer**

<https://github.com/chainer/chainer>

<https://chainer.org>

"Chainer is a Python-based deep learning framework aiming at flexibility. It provides automatic differentiation APIs based on the define-by-run approach (a.k.a. dynamic computational graphs) as well as object-oriented high-level APIs to build and train neural networks.

Chainer is a Python-based deep learning framework aiming at flexibility. It provides automatic differentiation APIs based on the define-by-run approach, also known as dynamic computational graphs, as well as object-oriented high-level APIs to build and train neural networks. It supports CUDA and cuDNN using CuPy for high performance training and inference.

Chainer is an open-source neural network framework with a Python API, whose core team of developers work at Preferred Networks, a machine-learning startup based in Tokyo drawing its engineers largely from the University of Tokyo."

**CNTK**

<https://github.com/Microsoft/cntk>

<https://docs.microsoft.com/en-us/cognitive-toolkit/>

The Microsoft Cognitive Toolkit (CNTK) is a unified deep learning toolkit that describes neural networks as a series of computational steps via a directed graph.

The library includes feed-forward DNNs, convolutional nets and recurrent networks.

**Deeplearning.scala**

<https://github.com/ThoughtWorksInc/DeepLearning.scala>

<https://skymind.ai/wiki/scala-ai>

DeepLearning.scala is a simple library for creating complex neural networks from object-oriented and functional programming constructs.

ThoughtWorks built a simple library for creating complex neural networks called Deeplearning.scala, which uses Skymind’s scientific computing library ND4J.

**Deeplearning4j**

<https://github.com/deeplearning4j/deeplearning4j/tree/master/deeplearning4j>

<https://deeplearning4j.org/docs/latest/deeplearning4j-scaleout-intro>

<https://github.com/deeplearning4j/deeplearning4j/tree/master/scalnet>

Eclipse Deeplearning4j, which is distinguished from other frameworks in its API languages, intent and integrations. DL4J is a JVM-based, industry-focused, commercially supported, distributed deep-learning framework that solves problems involving massive amounts of data in a reasonable amount of time. It integrates with Kafka, Hadoop and Spark using an arbitrary number of GPUs or CPUs, and it has a number you can call if anything breaks. "Deeplearning4j supports neural network training on a cluster of CPU or GPU machines using Apache Spark. Deeplearning4j also supports distributed evaluation as well as distributed inference using Spark.

Spark is not always the most appropriate tool for training neural networks.

You should use Spark when:

You have a cluster of machines for training (not just a single machine - this includes multi-GPU machines)

You need more than single machine to train the network

Your network is large to justify a distributed implementation" In speed, DL4J performance is equal to Caffe on non-trivial image-processing tasks on multiple GPUs, and faster than Tensorflow or Torch.

"A Scala wrapper for Deeplearning4j, inspired by Keras. Scala + DL + Spark + GPUs ScalNet is the Keras-like Scala API for Deeplearning4j"

**Dlib**

<https://github.com/davisking/dlib>

<http://dlib.net>

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.

No other packages are required to use the library. Only APIs that are provided by an out of the box OS are needed.

**DSSTNE**

<https://github.com/amzn/amazon-dsstne>

<https://aws.amazon.com/blogs/big-data/generating-recommendations-at-amazon-scale-with-apache-spark-and-amazon-dsstne/>

DSSTNE (pronounced "Destiny") is an open source software library for training and deploying recommendation models with sparse inputs, fully connected hidden layers, and sparse outputs. Amazon’s Deep Scalable Sparse Tensor Network Engine, or DSSTNE, is a library for building models for machine- and deep learning. It is one of the more recent of many open-source deep-learning libraries to be released, after Tensorflow and CNTK, and Amazon has since backed MxNet with AWS, so its future is not clear. Written largely in C++, DSSTNE appears to be fast, although it has not attracted as large a community as the other libraries.

**Dynet**

<https://github.com/clab/dynet>

The Dynamic Neural Network Toolkit DyNet is a neural network library developed by Carnegie Mellon University and many others. It is written in C++ (with bindings in Python) and is designed to be efficient when run on either CPU or GPU, and to work well with networks that have dynamic structures that change for every training instance.

**Fast.ai**

<https://github.com/fastai/fastai>

The fastai library simplifies training fast and accurate neural nets using modern best practices. The library is based on research into deep learning best practices undertaken at fast.ai, and includes "out of the box" support for vision, text, tabular, and collab (collaborative filtering) models.

**Gluon**

<https://github.com/gluon-api/gluon-api/>

The Gluon API specification is an effort to improve speed, flexibility, and accessibility of deep learning technology for all developers, regardless of their deep learning framework of choice. in October 2017 — Microsoft and Amazon’s AWS jointly announced the Gluon API. Gluon is a high-level Python deep learning interface that wraps MXNet and soon it will also include Microsoft’s CNTK. Gluon is a direct competitor for Keras and although AWS claims that they strongly support all deep learning frameworks they, of course, bet on Gluon for the democratization of AI. Named after a subatomic particle, Gluon is an API over Amazon’s MxNet that was introduced by Amazon and Microsoft in October 2017. It will also integrate with Microsoft’s CNTK. While it is similar to Keras in its intent and place in the stack, it is distinguished by its dynamic computation graph

**Keras**

<https://github.com/keras-team/keras>

<https://medium.com/tensorflow/standardizing-on-keras-guidance-on-high-level-apis-in-tensorflow-2-0-bad2b04c819a>

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. "Keras is a high-level deep learning API, written in Python and created by François Chollet — a deep learning researcher at Google. Google announced in 2017 that Keras has been chosen to serve as the high-level API of TensorFlow. This means that Keras will be included in the next TensorFlow release. Next to TensorFlow, Keras can also use Theano or CNTK as backend.

Keras is powerful because it’s really straightforward to create a deep learning model by stacking multiple layers. When using Keras, the user doesn’t have to do the maths behind the layers." "Works with Theano, TensorFlow and Deeplearning4j backends (CNTK backend to come)

Keras is a higher-level API with a configurable back-end. At the moment TensorFlow, Theano and CNTK are supported, though perhaps in the not too distant future PyTorch will be included as well. Keras is also distributed with TensorFlow as a part of tf.contrib." The Keras API has implementations for TensorFlow, MXNet, TypeScript, JavaScript, CNTK, Theano, PlaidML, Scala, CoreML, and other libraries.

**Ludwig**

<https://github.com/uber/ludwig>

Ludwig is a toolbox built on top of TensorFlow that allows to train and test deep learning models without the need to write code. All you need to provide is a CSV file containing your data, a list of columns to use as inputs, and a list of columns to use as outputs, Ludwig will do the rest.

**minimaxir**

<https://github.com/minimaxir/automl-gs>

offers a zero code/model definition interface to getting an optimized model and data transformation pipeline in multiple popular ML/DL frameworks, with minimal Python dependencies (pandas + scikit-learn + your framework of choice). "supports

TensorFlow (via tf.keras) | tensorflow

XGBoost (w/ histogram binning) | xgboost

Generates native Python code; no platform lock-in, and no need to use automl-gs after the model script is created."

**MXNet**

<https://github.com/apache/incubator-mxnet>

<https://mxnet.apache.org/>

Lightweight, Portable, Flexible Distributed/Mobile Deep Learning with Dynamic, Mutation-aware Dataflow Dep Scheduler; for Python, R, Julia, Scala, Go, Javascript and more

Another popular deep learning framework is MXNet, supported by Microsoft and Amazon. MXNet have been around for a while, but when MXNet is mentioned as deep learning framework, I often hear people respond with “isn’t that a deep learning framework for R?”. Yes it is, but it’s more. It actually supports many languages, from C++ to Python, JavaScript, Go, and, indeed, R. Where MXNet stands-out is its scalability and performance (stay tuned for Part II — where we will compare the most popular frameworks on speed amongst other metrics).

"symbolic programming and imperative programming to maximize efficiency and productivity.

In its core is 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. The library is portable and lightweight, and it scales to multiple GPUs and multiple machines."

MxNet is a machine-learning framework with APIs is languages such as R, Python and Julia which has been adopted by Amazon Web Services. Parts of Apple are also rumored to use it after the company’s acquisition of Graphlab/Dato/Turi in 2016. A fast and flexible library, MxNet involves Pedro Domingos and a team of researchers at the University of Washington.

"Apache MXNet (incubating) is a deep learning framework designed for both efficiency and flexibility. It allows you to mix symbolic and imperative programming to maximize efficiency and productivity. At its core, MXNet contains a dynamic dependency scheduler that automatically parallelizes both symbolic and imperative operations on the fly. A graph optimization layer on top of that makes symbolic execution fast and memory efficient. MXNet is portable and lightweight, scaling effectively to multiple GPUs and multiple machines.

MXNet is more than a deep learning project. It is a collection of blue prints and guidelines for building deep learning systems, and interesting insights of DL systems for hackers."

**PaddlePaddle**

<https://github.com/PaddlePaddle/Paddle>

PaddlePaddle (PArallel Distributed Deep LEarning) is an easy-to-use, efficient, flexible and scalable deep learning platform, which is originally developed by Baidu scientists and engineers for the purpose of applying deep learning to many products at Baidu. "PaddlePaddle provides an intuitive and flexible interface for loading data and specifying model structures. It supports CNN, RNN, multiple variants and configures complicated deep models easily.

It also provides extremely optimized operations, memory recycling, and network communication. PaddlePaddle makes it easy to scale heterogeneous computing resources and storage to accelerate the training process."

"Paddle is a deep-learning framework created and supported by Baidu. Its name stands for PArallel Distributed Deep LEarning. Paddle is the most recent major framework to be released, and like most others, it offers a Python API."

**PyTorch**

<https://github.com/pytorch/pytorch>

<https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750>

<https://pytorch.org>

Tensors and Dynamic neural networks in Python with strong GPU acceleration. "PyTorch was introduced by Facebook, amongst others, in January 2017. It’s a port to the popular Torch framework (implemented in C with a wrapper in Lua) with the Torch binaries wrapped in GPU accelerated Python.

Next to the GPU acceleration and the efficient usages of memory, the main driver behind the popularity of PyTorch is the use of dynamic computational graphs. These dynamic computational graphs were already being used by other, lesser known, deep learning frameworks like Chainer. The advantage of these dynamic graphs is that the graphs are defined by the run (“define by run”) instead of the traditional “define and run”. Especially, in cases where the input can vary, for example with unstructured data like text, this is extremely useful and efficient."

"PyTorch is a Python package that provides two high-level features:

Tensor computation (like numpy) with strong GPU acceleration

Deep Neural Networks built on a tape-based autograd system

You can reuse your favorite Python packages such as numpy, scipy and Cython to extend PyTorch when needed."

PyTorch is better for rapid prototyping in research, for hobbyists and for small scale projects. TensorFlow is better for large-scale deployments, especially when cross-platform and embedded deployment is a consideration.

**Sonnet**

<https://github.com/deepmind/sonnet>

<https://sonnet.dev/>

Sonnet is a library built on top of TensorFlow for building complex neural networks.

in 2017, Google’s DeepMind released Sonnet (a high-level object oriented library build on top of TensorFlow

**SparkFlow**

<https://github.com/lifeomic/sparkflow>

<https://medium.com/lifeomic/sparkflow-train-tensorflow-models-with-apache-spark-pipelines-74dca32f60f3>

This is an implementation of Tensorflow on Spark. The goal of this library is to provide a simple, understandable interface in using Tensorflow on Spark. With SparkFlow, you can easily integrate your deep learning model with a ML Spark Pipeline.

"LifeOmic released SparkFlow in 2018. SparkFlow utilizes the convenient interface from Spark’s pipeline api and combines it with TensorFlow. It can be downloaded from Github or installed through pip, using “pip install sparkflow.”

Although other open-source libraries exist to train TensorFlow models on Apache Spark, very few take advantage of SparkML’s biggest machine learning strength, which is integrating deep learning models with pipelines." SparkFlow takes advantage of Spark’s driver/executor architecture, using the driver as a parameter server and the executors as workers. Naturally, this creates a simple path to implement asynchronous training algorithms such as Hogwild. The Hogwild algorithm parallelizes the training of neural networks by having a neural network live on the master node, and copies of that network on each of the task instances. The task instances then compute the gradients on a batch of data, and send them to the driver for updating the weights. Hogwild is different from asynchronous training as it prescribes a lock free approach in updating the master network gradients. (As a side note, SparkFlow supports both asynchronous training with locks and lock-free approaches).

**TensorFlow**

<https://github.com/tensorflow/tensorflow>

<https://www.bizety.com/2018/10/29/tensorflow-vs-caffe/>

<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/distribute>

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

"TensorFlow, all the necessary adjustments are performed via the tf.device(), in which one designates the use of GPUs.

No further documentation is necessary, nor are further changes to the API. TensorFlow also offers a more flexible architecture as you can run two copies of a model on two GPUs, or a single large model over two GPUs.

support for multiple machines is a given." TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. For visualizing TensorFlow results, TensorFlow offers TensorBoard, suite of visualization tools. Google created TensorFlow to replace Theano. The two libraries are in fact quite similar. Some of the creators of Theano, such as Ian Goodfellow, went on to create Tensorflow at Google before leaving for OpenAI. "TensorFlow provides stable Python and C APIs as well as non-guaranteed backwards compatible API's for C++, Go, Java, JavaScript, and Swift.

Reflecting these rapid changes, we have started work on the next major version of TensorFlow. TensorFlow 2.0 will be a major milestone, with a focus on ease of use. Here are some highlights of what users can expect with TensorFlow 2.0:

Eager execution will be a central feature of 2.0. It aligns users’ expectations about the programming model better with TensorFlow practice and should make TensorFlow easier to learn and apply.

Support for more platforms and languages, and improved compatibility and parity between these components via standardization on exchange formats and alignment of APIs.

We will remove deprecated APIs and reduce the amount of duplication, which has caused confusion for users."

"TensorFlow includes an implementation of the Keras API (in the tf.keras module) with TensorFlow-specific enhancements. These include support for eager execution for intuitive debugging and fast iteration, support for the TensorFlow SavedModel model exchange format, and integrated support for distributed training, including training on TPUs.

**Theano**

<https://github.com/Theano/Theano>

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. It can use GPUs and perform efficient symbolic differentiation. The first mainstream deep learning framework was called Theano and was created and maintained by MILA and led by Yoshua Bengio, a deep learning pioneer. However, work on Theano came to a halt on Sept. 28, 2017 after one final version was released. It was not a surprising decision as so many other deep learning frameworks had sprung into being, usually open source Python and frequently backed by one of the big tech companies like Google and Microsoft.

**Torch**

<https://github.com/torch/torch7>

<https://awni.github.io/pytorch-tensorflow/>

Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation. http://torch.ch/

**Other References:**

<https://skymind.ai/wiki/comparison-frameworks-dl4j-tensorflow-pytorch>

<https://developer.nvidia.com/deep-learning-frameworks>

<https://www.kdnuggets.com/2018/04/top-16-open-source-deep-learning-libraries.html>

<https://www.quora.com/Are-all-programming-languages-based-on-C>

**Considered but not included:**

**ONNX** - model format only

"Announced in September of 2017 and the release of V1 in December, ONNX is an open format to represent deep learning models. This allows users to more easily move models between different frameworks. For example, it allows you to build a PyTorch model and run the model for inference using MXNet.

ONNX is launched by Microsoft, AWS, and Facebook amongst others. It doesn’t come as a surprise that Google isn’t part of this list. ONNX supports Caffe2, Microsoft Cognitive Toolkit, MXNet, and PyTorch from the start, but like with other open source projects the community already added a converter for TensorFlow as well."

**Matlab** - not open source

MATLAB makes deep learning easy for engineers, scientists and domain experts. With tools and functions for managing and labeling large data sets, MATLAB also offers specialized toolboxes for working with machine learning, neural networks, computer vision, and automated driving. With just a few lines of code, MATLAB lets you create and visualize models, and deploy models to servers and embedded devices without being an expert. MATLAB also enables users to generate high-performance CUDA code for deep learning and vision applications automatically from MATLAB code.

**SKIL** - pretty sure this is just a DS platform and not an API for DL

<https://docs.skymind.ai/>

The community edition of the Skymind Intelligence Layer (SKIL) is free. It takes data science projects from prototype to production quickly and easily. SKIL bridges the gap between the Python ecosystem and the JVM with a cross-team platform for Data Scientists, Data Engineers, and DevOps/IT. It is an automation tool for machine-learning workflows that enables easy training on Spark-GPU clusters, experiment tracking, one-click deployment of trained models, model performance monitoring and more.

**Deep Learning Pipelines** - scoring only

<https://github.com/databricks/spark-deep-learning>

Deep Learning Pipelines provides high-level APIs for scalable deep learning in Python with Apache Spark. "The library comes from Databricks and leverages Spark for its two strongest facets:

In the spirit of Spark and Spark MLlib, it provides easy-to-use APIs that enable deep learning in very few lines of code.

It uses Spark's powerful distributed engine to scale out deep learning on massive datasets.

Currently, TensorFlow and TensorFlow-backed Keras workflows are supported, with a focus on:

large-scale inference / scoring

transfer learning and hyperparameter tuning on image data

Furthermore, it provides tools for data scientists and machine learning experts to turn deep learning models into SQL functions that can be used by a much wider group of users. It does not perform single-model distributed training - this is an area of active research, and here we aim to provide the most practical solutions for the majority of deep learning use cases." 2017 release by DBricks

**Horovod** - a separate framework for distributing graphs

<https://github.com/horovod/horovod>