# Comparison of deep learning software

**Some of Known Deep learning software**

1. Tensorflow
2. Theano
3. Caffe
4. Torch
5. Apache Singa
6. Chainer
7. Deeplearning4j
8. DyNet(Don’t support parallel execution (multi nodes))
9. Keras
10. Mxnet
11. Knet.jl
12. Microsoft Cognitive Toolkit - CNTK
13. TensorLayer
14. OpenNN(Don’t support CUDA, RNN nets, Convolution nets)
15. Neural Designer(Don’t support CUDA, RNN nets, Convolution nets)
16. Lasagne
17. Wolfram Mathematica (Don’t support RNN nets)
18. Dato GraphLab
19. H2o.ai

# **Criteria**

In order to determine if each feature is considered as a strength or a weakness, criteria must be defined. These should differ depending on needs, especially the first criterion:

* Works on Windows 10 to avoid dual boot or virtual machine
* Can be used by a neophyte
* Uses GPU for faster training
* Can model different types of networks to do experiments
* Uses an “easy” programming language to create prototypes faster
* Has pretrained models to avoid wasting time training on classic datasets
* Free

[**TensorFlow**](https://en.wikipedia.org/wiki/TensorFlow)

**Interface**: Python, (C/C++public API only for executing graphs)

**Platform**: Linux, Mac OS X,Windows

* This library would improve the portability of machine learning so that research models could easily applied to commercial-grade application.
* Based on the concept of a computational graph.
* Tensorflow users have to use Keras as a simplified interface to TensorFlow.
* Tensorboard a visualization tool for network architecture and performance, it view different summary-level metrics and changes over time throughout the training process.
* Free open online course on udacity <https://classroom.udacity.com/courses/ud730/lessons/6370362152/concepts/63798118170923#>

**Pros and Cons**

* (+) Python + Numpy
* (+) Computational graph abstraction, like Theano
* (+) Faster compile times than Theano
* (+) TensorBoard for visualization
* (+) Data and model parallelism
* (-) Slower than other frameworks
* (-) Much “fatter” than Torch; more magic
* (-) Not many pretrained models
* (-) Computational graph is pure Python, therefore slow
* (-) No commercial support
* (-) Drops out to Python to load each new training batch
* (-) Not very toolable
* (-) Dynamic typing is error-prone on large software projects

**Caffe**[**[2]**](https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software#cite_note-caffe-3)

**Interface:**

C++, command line, Python,MATLAB[7]

**Platform:**

Linux, Mac OS X,AWS,unofficial Android port,Windows support by Microsoft Research,unofficial Windows port

Pros and Cons:

* (+)Good for machine vision or a for casting application.
* (+) Good for feedforward networks and image processing
* (+) Good for finetuning existing networks
* (+) Train models without writing any code
* (+) Python interface is pretty useful
* (-) Need to write C++ / CUDA for new GPU layers
* (-) Not good for recurrent networks
* (-) Cumbersome for big networks (GoogLeNet, ResNet)
* (-) Not extensible, bit of a hairball
* (-) No commercial support

[**Torch**](https://en.wikipedia.org/wiki/Torch_(machine_learning))

**interfaces:**Lua, LuaJIT,C, utility library for C++/OpenCL

**Platform** :Linux, Mac OS X,Windows,Android, iOS

Pros and Cons:

* (+) Very good overall, nice architecture, can model lots of network types, fast
* (+/-) Uses Lua which is a nice language but not widly used
* (-) Does not work on Windows

[**Theano**](https://en.wikipedia.org/wiki/Theano_(software))

**interface:**[Python](https://en.wikipedia.org/wiki/Python_(programming_language))

**Platform**:[Cross-platform](https://en.wikipedia.org/wiki/Cross-platform)

* (+) Python + Numpy
* (+) Computational graph is nice abstraction
* (+) RNNs fit nicely in computational graph
* (-) Raw Theano is somewhat low-level
* (+) High level wrappers (Keras, Lasagne) ease the pain
* (-) Error messages can be unhelpful
* (-) Large models can have long compile times
* (-) Much “fatter” than Torch
* (-) Patchy support for pretrained models
* (-) Buggy on AWS

[**MXNet**](https://en.wikipedia.org/wiki/MXNet)

**Interface**: C++, Python,Julia, Matlab,JavaScript, Go,R, Scala  
**Platform**:Linux, Mac OS X,Windows,AWS, Android,iOS, JavaScript

[MxNet](https://github.com/dmlc/mxnet) is a machine-learning framework with APIs is languages such as R, Python and Julia which has been[adopted by Amazon Web Services](http://www.allthingsdistributed.com/2016/11/mxnet-default-framework-deep-learning-aws.html). 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. A [comparison](https://deeplearning4j.org/mxnet) between MxNet and some aspects of Deeplearning4j can be found here.

**Keras**

[Keras](https://deeplearning4j.org/keras.io) is a deep-learning library that sits atop Theano and TensorFlow, providing an intuitive API inspired by Torch. Perhaps the best Python API in existence. Deeplearning4j [imports models from Keras](https://deeplearning4j.org/keras). It was created by [Francois Chollet](https://twitter.com/fchollet), a software engineer at Google.

[**Deeplearning4j**](https://en.wikipedia.org/wiki/Deeplearning4j)

**Interface :**Java, Scala,Clojure

**Platform :**Linux, Mac OS X,Windows, Android(Cross-platform)

it is distinguished from its predecessors in both programming language and intent. DL4J is a JVM-based, industry-focused, commercially supported, distributed deep-learning framework intended to solve problems involving massive amounts of data in a reasonable amount of time. It integrates with Hadoop and [Spark](https://deeplearning4j.org/spark) using an arbitrary number of [GPUs](https://deeplearning4j.org/gpu) or [CPUs](https://deeplearning4j.org/native), and it has [a number you can call](http://www.skymind.io/contact?__hstc=3042607.45b343cdfab3de5ca9fb0e7f1b161b33.1485715211850.1485715211850.1485715211850.1&__hssc=3042607.1.1485715211851&__hsfp=2449015499) if anything breaks. DL4J is portable and platform neutral, rather than being tied to any cloud service such as AWS, Azure or Google Cloud. In speed, its [performance is equal to Caffe](https://github.com/deeplearning4j/dl4j-benchmark) on non-trivial image-processing tasks on multiple GPUs, and better than Tensorflow or Torch.

**Pros and cons:**

* (+) Very good overall, nice architecture, can model lots of network types, fast
* (+/-) Seems to be oriented toward business
* (+/-) Use of Java great idea but it makes prototyping slower

**Finally I think Theano and caffe not suitable for us.**

**Reference**  
<https://deeplearning4j.org/compare-dl4j-torch7-pylearn.html>

<https://www.tensorflow.org/>

<http://ankivil.com/choosing-a-deep-learning-software/>

<https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software>