Clermont-Auvergne University

IT and Technology Report

1ère Année de Master Informatique

By

Alexis PLESSIER

Date : 03/22/2021

I – TensorFlow

TensorFlow is an open source machine learning tool developed by Google and released under the Apache license in 2015. It is written in C++ and Python and available on all platforms, making it one of the most widely used tools in the AI field. TensorFlow is capable of running deep neural networks, handwriting classification, image recognition (used by the Pentagon, for example), sequence modeling and language processing. It is also capable of predicting results on different models.

TensorFlow allows developers to create data flow graphs - structures that describe how data moves through a graph, or series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor. All this is possible thanks to its implementation in Python because of its objects but being transformed into binary form in C++ to be more efficient.

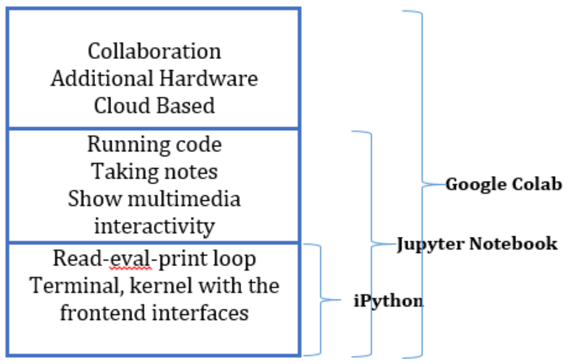
Its greatest asset is also its simplicity, the user can concentrate on his model and TensorFlow will take care of the details and any other parameters. In direct competition, there are other Machine Learning frameworks that have all their advantages and disadvantages such as PyTorch, CNTK or ApacheMXNet.

II – Google Colab

Google Colab is the equivalent of a NoteBook jupiter (.ipynb) that runs in the Google cloud and is completely free. It requires no installation whatsoever and allows simultaneous sharing and editing. It is best known for supporting various Machine Learning libraries that are rather easy to load into Google Colab.

It allows you to write/load and save NoteBooks or simply python code in the Google Drive but also has a direct link to GitHub where you can import/export projects. It also allows you to import your own datasets and an integration of PyTorch, TensorFlow, Keras or OpenCSV, all practical tools for Machine Learning. One of the big advantages is that it allows us to use the dedicated Google Colab GPU and its accelerator which can make the difference in some projects requiring more computation.

Here is the major difference between Google Colab and its competitors:

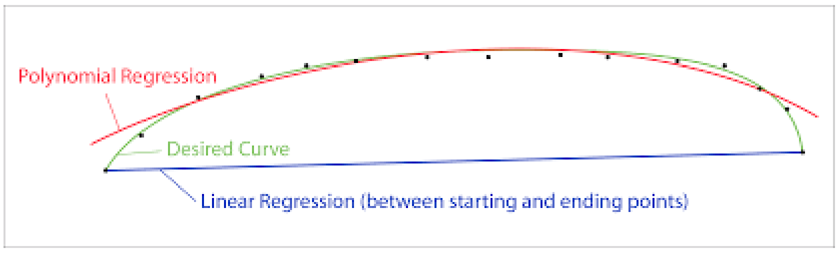


III – Curve Fitting

Curve fitting refers to a type of optimisation that sets up a set of optimal parameters for a function defined to best match its observations. Unlike supervised learning, Curve Fitting requires a function that has known inputs and outputs. The function can take many forms such as a line (linear regression), a curved line (polynomial regression) or others.

Let's imagine that we have a graph with a scatter plot. We would like to find a function y that would best approximate all the points present. To do this, we will try to reduce the error by changing our parameters. The error is calculated as the difference between the output of our function and the output observed on the graph for the same input.

To get close to the most likely function, we need to detect a trend such as a straight, curved, sinusoidal line, ... The parameters are then determined according to the function of the probable curve (Example: y=a1x1+a2x2+a3x3+b for 3 points and a linear curve, here we are looking for a and b).



(here, the linear regression is useless because it is too far from the model

and the points we have, so polynomial regression)

IV – TensorFlow Callbacks

The TensorFlow Callback is a function or group of code that runs at a specific time when training our model in TensorFlow.

With models becoming more complex and resource intensive, the time to train them has increased greatly in recent years, and it is normal that it can take several hours. After setting the training parameters of interest (learning rate, optimisers, loss, ...), the model is trained. From this moment on, there is no way to stop the process or even to modify the parameters until the model has finished. It may happen that after hours we want to change parameters as the model evolves. Callbacks are used for this, i.e. at a well-defined moment in the learning process, we have the choice of modifying parameters or otherwise or letting it continue. This can be very useful to correct an error on our part or to implement a new directive.

V – Image Classification

Image classification is a simple thing for a human being but it quickly becomes complicated for a machine and has quickly become an important discipline like computer vision. There are potentially as many classes as there are images in a training dataset and classifying them by hand would be a tedious task so it was decided to automate them. Some examples would be cancer detection on X-ray or determining the number that is written.

A classification is characterised by 4 major steps:

1. Image preparation : The image often needs to be reworked because it has shape or colour flaws and may mislead the model being trained. This is also used to have images with similar parameters and therefore to highlight possible differences.

2. Object detection : Detects an object in the image and determines its position in order to identify it.

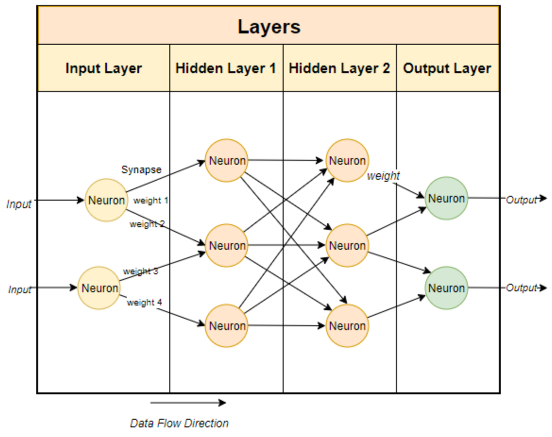
3. Parameter extraction and training : We look for the most accurate and unique parameters that can refer to an existing image class and match the image to help the model differentiate classes in the future.

4. Object classification : Once the model has been trained, it will be applied to the image and will apply the parameters of each class to determine which class the image is most likely to belong to.

VI – Weight Comparaison

Weights and biases are model variables that are updated to improve the accuracy of the network. A weight is applied to the input of each neuron to calculate an output.

Neural networks update these weights continuously. There is therefore a feedback loop implemented in most neural networks.



VII – Transfer Learning

Transfer Learning refers to the set of methods that allow the transfer of knowledge gained from solving a given problem to another problem.

The models used in this field often require high computing times and significant resources. However, by using pre-trained models as a starting point, Transfer Learning allows to quickly develop efficient models and to efficiently solve complex problems in Computer Vision or Natural Language Processing, NLP.

Transfer Learning is based on the simple idea of re-using knowledge acquired in other configurations (sources) to solve a particular problem (target). In this context, one can distinguish several approaches depending on what one wants to transfer, when and how the transfer is carried out.

The architecture of Deep Learning models is very often in the form of a stack of layers of neurons. These layers learn different characteristics depending on the level at which they are located. The last layer (usually a fully connected layer, in the case of supervised learning) is used to obtain the final output. The idea here is to add an output for example. Instead of re-training the whole model with an extra output, we will use the already trained model and add our output.

VIII - Time series forecasting with LSTM

LSTM (Long Short-Term Memory) is an architecture based on recurrent neural networks (RNNs) that is widely used in natural language processing and time series forecasting.

LSTM addresses a huge problem that recurrent neural networks suffer from: short memory. Using a series of "gates", each with its own RNN, LSTM is able to retain, forget or ignore data points based on a probabilistic model.

LSTMs also help solve explosive gradient and evanescent gradient problems. In simple terms, these problems are the result of repeated adjustments of weights when training a neural network. With repeated epochs, the gradients become larger or smaller, and with each adjustment, it becomes easier for the gradients in the network to compound in both directions. This compounding makes the gradients either much too large or much too small. While the explosion and disappearance of gradients are major drawbacks of using traditional RNNs, the LSTM architecture greatly mitigates these problems.

Once a prediction is made, it is fed back into the model to predict the next value in the sequence. With each prediction, an error is introduced into the model. To avoid gradient explosion, the values are "squashed" by (usually) sigmoid and tanh activation functions before entering and leaving the gate.