**Implementing DNN Classifier with TensorFlow and Keras**

**TensorFlow** as being described in its official website 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.

So, What is Keras?

**Keras** is a high-level API for Deep Learning that can implement (build, train and evaluate) any sort of Deep Neural Networks (DNN). It was developed as a research project by Francois Chollet. It becomes popular because of its flexibility, its ease of use and its ability to do NN heavy computations.

TensorFlow adopted Keras as its official high-level API and it now comes bundled with its own Keras implementation, tensorflow.keras. It only supports TensorFlow as the backend, but it has the advantage of offering some very useful extra features.

**Installation**:

It's preferred to build your TensorFlow backend library on a virtual environment. You can build it directly on your conda-base environment or create a new environment with the folowing command:

$ conda create -n tf

The activate it:

$ conda activate tf

Then to install TensorFlow (latest version) just run the following command:

$ pip install tensorflow

If you want to install a separate copy of Keras:

$ pip install keras

**Preparing the image data:**

Before training, we’ll preprocess the data by:

1. Scaling it (as we will use the Gradient Descent to train our network) so that all values are in the [0, 1] interval.

2. Training images, for instance, were stored in an array of shape (60000, 28, 28) of type uint8 with values in the [0, 255] interval per pixel. We will transform it into a float32 array of shape (60000, 28 \* 28) with values between 0 and 1.

Dataset link: (<https://github.com/zalandoresearch/fashion-mnist>)

[Fashion MNIST (kaggle.com)](https://www.kaggle.com/datasets/zalando-research/fashionmnist/data)

* Create Sequential Model
* Create checkpoint to save model at the end of each epoch
* Create Early Stopping checkpoint
* Evaluate the model and plot the results
* Test your model