# Tìm hiểu tổng quan về TensorFlow và Keras

## Keras

Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.

Keras is relatively easy to learn and work with because it provides a python frontend with a high level of abstraction while having the option of multiple back-ends for computation purposes. This makes Keras slower than other deep learning frameworks, but extremely beginner-friendly.

Keras allows you to switch between different back ends. The frameworks supported by Keras are:

* Tensorflow
* Theano
* PlaidML
* MXNet
* CNTK (Microsoft Cognitive Toolkit)

Out of these five frameworks, TensorFlow has adopted Keras as its official high-level API. Keras is embedded in TensorFlow and can be used to perform deep learning fast as it provides inbuilt modules for all neural network computations. At the same time, computation involving tensors, computation graphs, sessions, etc can be custom made using the Tensorflow Core API, which gives you total flexibility and control over your application and lets you implement your ideas in a relatively short time.

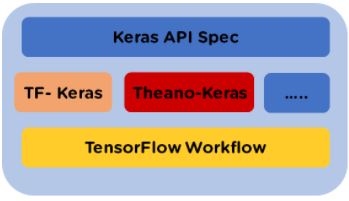


Figure 1: Keras backend

* Keras is an API that was made to be easy to learn for people. Keras was made to be simple. It offers consistent & simple APIs, reduces the actions required to implement common code, and explains user error clearly.
* Prototyping time in Keras is less. This means that your ideas can be implemented and deployed in a shorter time. Keras also provides a variety of deployment options depending on user needs.
* Languages with a high level of abstraction and inbuilt features are slow and building custom features in then can be hard. But Keras runs on top of TensorFlow and is relatively fast. Keras is also deeply integrated with TensorFlow, so you can create customized workflows with ease.
* The research community for Keras is vast and highly developed. The documentation and help available are far more extensive than other deep learning frameworks.
* Keras is used commercially by many companies like Netflix, Uber, Square, Yelp, etc which have deployed products in the public domain which are built using Keras.

Apart from this, Keras has features such as :

* It runs smoothly on both CPU and GPU.
* It supports almost all neural network models.
* It is modular in nature, which makes it expressive, flexible, and apt for innovative research.

## Tensorflow

TensorFlow is an open-source library that the Google Brain team developed in 2012. Python is by far the most common language that TensorFlow uses. You can import the TensorFlow library into your Python environment and perform in-depth learning development.

There is a sure way in which the program gets executed. You first create nodes, which process- the data in the form of a graph. The data gets stored in the form of tensors, and the tensor data flows to various nodes.

One of TensorFlow’s best qualities is that it makes code development easy. The readily available APIs save users from rewriting some of the code that would otherwise have been time-consuming. TensorFlow speeds up the process of training a model. Additionally, the chances of errors in the program are also reduced, typically by 55 to 85 percent.

The other important aspect is TensorFlow is highly scalable. You can write your code and then make it run either on CPU, GPU, or across a cluster of these systems for the training purpose.

Generally, training the model is where a large part of the computation goes. Also, the process of training is repeated multiple times to solve any issues that may arise. This process leads to the consumption of more power, and therefore, you need a distributed computing. If you need to process large amounts of data, TensorFlow makes it easy by running the code in a distributed manner.

GPUs, or graphical processing units, have become very popular. Nvidia is one of the leaders in this space. It is good at performing mathematical computations, such as matrix multiplication, and plays a significant role in deep learning. TensorFlow also has integration with C++ and Python API, making development much faster.

# 2) Trình bày chi tiết về các class sau trong Keras:

## a) Dense

The dense layer is a neural network layer that is connected deeply, which means each [neuron](https://machinelearningknowledge.ai/glossary/artificial-neuron/) in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models.

In the background, the dense layer performs a matrix-vector multiplication. The values used in the matrix are actually parameters that can be trained and updated with the help of backpropagation.

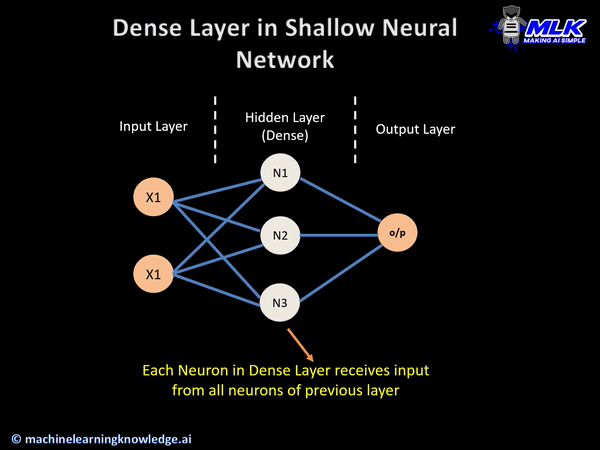
The output generated by the dense layer is an ‘m’ dimensional vector. Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also applies operations like rotation, scaling, translation on the vector.

**output = activation(dot(input, kernel) + bias)**

In the above equation, **activation** is used for performing **element-wise activation** and the **kernel** is the **weights matrix** created by the layer, and **bias** is a bias vector created by the layer.

Keras dense layer on the output layer performs **dot product** of **input tensor** and **weight kernel matrix**.

A bias vector is added and element-wise activation is performed on output values.



## b) Model

Arguments:

* **inputs:** It can be defined as an input that is being fed to the model. It can either be an object of **Input** or a list of objects, i.e., keras.Input.
* **outputs:** It refers to the model's output.
* **name:** It can be a string that defines the model's name.

Following are the two ways by which the models can be instantiated:

I. In the first way, we will do with the help of "Functional API". We will start with the **Input** followed by connecting the layer calls for specifying the forward pass of the model and thus end by creating the model by utilizing the inputs as well as outputs.

import tensorflow as tf

inputs = tf.keras.Input(shape=(3,))

x = tf.keras.layers.Dense(4, activation=tf.nn.relu)(inputs)

outputs = tf.keras.layers.Dense(5, activation=tf.nn.softmax)(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)

II. In the second way we will do by subclassing the **Model** class. Here first we will define layers in **\_init\_** followed by executing the forward pass of the model in the **call**.

import tensorflow as tf

class MyModel(tf.keras.Model):

  def \_\_init\_\_(self):

    super(MyModel, self).\_\_init\_\_()

    self.dense1 = tf.keras.layers.Dense(4, activation=tf.nn.relu)

    self.dense2 = tf.keras.layers.Dense(5, activation=tf.nn.softmax)

  def call(self, inputs):

    x = self.dense1(inputs)

    return self.dense2(x)

model = MyModel()

While subclassing the **Model**, we can also have a **training** argument called Boolean (which is optional) in **the call** for specifying distinct behavior in inference as well as training:

import tensorflow as tf

class MyModel(tf.keras.Model):

  def \_\_init\_\_(self):

    super(MyModel, self).\_\_init\_\_()

    self.dense1 = tf.keras.layers.Dense(4, activation=tf.nn.relu)

    self.dense2 = tf.keras.layers.Dense(5, activation=tf.nn.softmax)

    self.dropout = tf.keras.layers.Dropout(0.5)

  def call(self, inputs, training=False):

    x = self.dense1(inputs)

    if training:

      x = self.dropout(x, training=training)

    return self.dense2(x)

model = MyModel()

After creating the model, we can config the model by incorporating losses and metrics by **model.compile()**. The model can be trained by using the model.fit() and with the help of **model.predict()** the model can make the predictions.