introduction

> TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.It is used for both research and production at Google.TensorFlow was developed by the Google Brain team for internal Google use.TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data.

Prerequisites for Tensorflow Tutorial

> You should have good knowledge of some programming language preferably Python. It is also important to have an understanding of machine learning to understand the use case and examples.Before directly understanding what is TensorFlow, you should know about deep learning and its libraries.Google built the underlying TensorFlow software with the C++ programming language. But in developing applications for this AI engine, coders can use either C++ or Python, the most popular language among deep learning researchers. The hope, however, is that outsiders will expand the tool to other languages, including Google Go, Java, and perhaps even Javascript, so that coders have more ways of building apps.

why use keras with tensorflow

> Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.Keras is an open-source software

> library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

1. Complexity

> Keras allows the development of models without the worry of backend details. While in TensorFlow you have to deal with computation details in the form of tensors and graphs.This feature of Keras provides more comfort and makes it less complex than TensorFlow.

2. Easy to Use API

> Keras is a high-level API. Keras uses either Tensorflow, Theano, or CNTK as its backend engines.Tensorflow provides both high and low-level APIs. Tensorflow is a math library that uses data flow programming for a wide variety of tasks.If you are looking for a neural network tool that is easy to use and has simple syntax then you will find Keras more favorable.

3. Fast development

> If you want to quickly deploy and test your deep learning models, choose Keras. Using Keras, you can create your models with very less lines of code and within a few minutes. Keras provides two APIs to write your neural network. These are: Model(functional API) Sequential With these APIs, you can easily create any complex neural network.

4. Performance

> Since Keras is not directly responsible for the backend computation, Keras is slower. Keras depends upon its backend engines for computation tasks. It provides an abstraction over its backend. To perform the underlying computations and training Keras calls its backend.On the other hand, Tensorflow is a symbolic math library. Its complex architecture focuses on reducing cognitive load for computation. Hence, Tensorflow is fast and provides high performance.

5. Functionality and Flexibility

> Tensorflow gives you more flexibility, more control, and advanced features for the creation of complex topologies. It provides more control over your network.Therefore if you want to define your own cost function, metric, or layer Or, if you want to perform operations on input weights or gradients, choose TensorFlow.

6. Dataset

> We prefer Keras if the size of the dataset is of relatively small or medium size. While if the dataset is large, we prefer TensorFlow because of fewer overheads. Also, TensorFlow provides more level of control, hence we have more options to handle large datasets.Tensorflow provides more number of inbuilt datasets than Keras. It contains all the datasets that are available in Keras and tf.datasets module of TensorFlow contains a wide range of dataset and these are classified under the following headings: Audio, Image, Image classification, object detection, question answering, structured, summarization, text, translate, and video. The datasets in Keras are present under the Keras.datasets module.

7. Debug

> Debugging the TensorFlow code is very difficult. In general, we perform de-bugging in TensorFlow debugger and done through the command line. We start by wrapping the TensorFlow session with,tf_debug.LocalCLIDebugWrapperSession(session) And then we execute the file with different necessary debug flags. Keras is high level and does not deal with backend computation, therefore debugging is easy. We can also check the output from each layer in Keras using keras.backend.function().

Installation

> Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. or

```
pip install tensorflow
```

```
pip install tensorflow-gpu
```

```
pip install Keras
```

CUDA is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

lets start

```
from google.colab import drive
drive.mount('/content/gdrive')
```

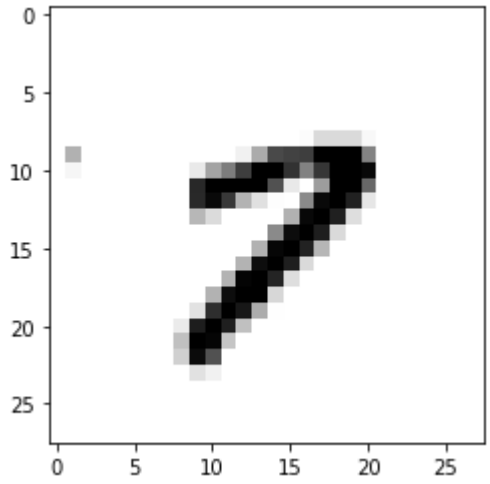
```
Mounted at /content/gdrive
```

```
from keras.datasets import mnist

# Load data
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Data attributes
print("train_images dimentions: ", train_images.ndim)
print("train_images shape: ", train_images.shape)
print("train_images type: ", train_images.dtype)
# Data visualization
import matplotlib.pyplot as plt
digit = train_images[29]
plt.imshow(digit, cmap='binary')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist
11493376/11490434 [==============================] - 0s 0us/step
train_images dimentions:  3
train_images shape:  (60000, 28, 28)
train_images type:  uint8
<matplotlib.image.AxesImage at 0x7f149c17c710>
```



2: load data from file example

```
import glob
import cv2
import numpy as np


images_path = "gdrive/MyDrive/data/CamVid/train/"
images = glob.glob(images_path + "*.png") + glob.glob(images_path + "*.jpg")
images.sort()
X = []
width = 200
height = 100
for img in images:
    image = cv2.imread(img)
    image = cv2.resize(image, (width, height))
    image = image / np.max(image)
    image = image.astype(np.float32)
    X.append(image)

# loading label images
labels_path = "gdrive/MyDrive/data/CamVid/trainannot/"
labels = glob.glob(labels_path + "*.png") + glob.glob(labels_path + "*.jpg")
labels.sort()




Y = []
out_width = 200
out_height = 100
nClasses = 12
seg_labels = np.zeros([out_height, out_width, nClasses], dtype='uint8')
for mask in labels:
    label = cv2.imread(mask)
    label = cv2.resize(label, (width, height))
    label = label[:,:,0]
    for c in range(nClasses):
        seg_labels[:,:,c] = (label == c)
    label = label.astype(np.uint8)
    Y.append(label)



print("X type: ", len(X))
print("Y shape: ", len(Y))
img_2 = X[90]
plt.imshow(img_2, cmap='binary')
```

```
X type:  380
Y shape:  367
<matplotlib.image.AxesImage at 0x7f1492cbfd30>
```