Tensorflow

Learn TF ~~Researchers~~ Engineer’s way

Ankit Agrawal, MTech CSE

About Tensorflow (TF)

- Developed by **Google brain** team
- One of the most popular framework for deep learning
  - Others are PyTorch, Microsoft’s CNTK etc
- Made open source by google back in 2015
- TF 2.0 was released in 2019
- Huge community support
- Horizontally scalable with negligible changes to code

Why TF

- With TF 2.0 its very easy to create deep learning models
- Developer friendly syntax
- Runs everywhere
    - Mobile (TF lite)
    - Browser (TF.js)
    - Large Production Environment (TF Extended)
- Supported across range of chipsets
    - CPU 😄
    - GPU
    - TPU
    - FSD Tesla (Full self driving computers)

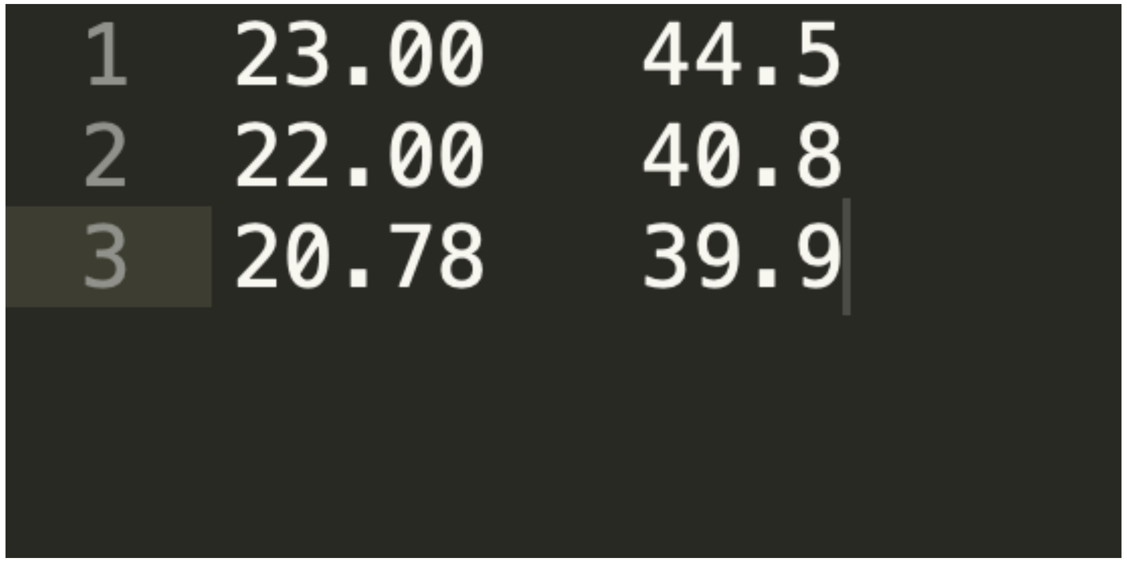
TF1.0 VS TF2.0

- TF 1.0 (Scary 👻, researcher oriented)
- TF 2.0 (Keras API’s has first hand support, Developer Friendly)
- You need not know 1.0 to learn 2.0
- What is Keras ?
  - It is open source high level API and can support multiple backends such as TF, Microsoft’s CNTK etc
  - Think of it as some high level programming language (Python) and TF 1.0 as low level language (Assembly)

Motivation

Why even bother about these things…

- It gives you super powers. You can predict the future (Just kidding…)
- But, it does gives you some power
- Let’s remember what we gained by learning programming
- You can program computers to do many things



```
23.00   44.5
22.00   40.8
20.78   39.9
```

- With the power of programming language we can reduce a tedious tasks which could have taken hours to seconds.
- Let’s see what TF framework has in its arsenal for us

Power of Deep Learning Frameworks

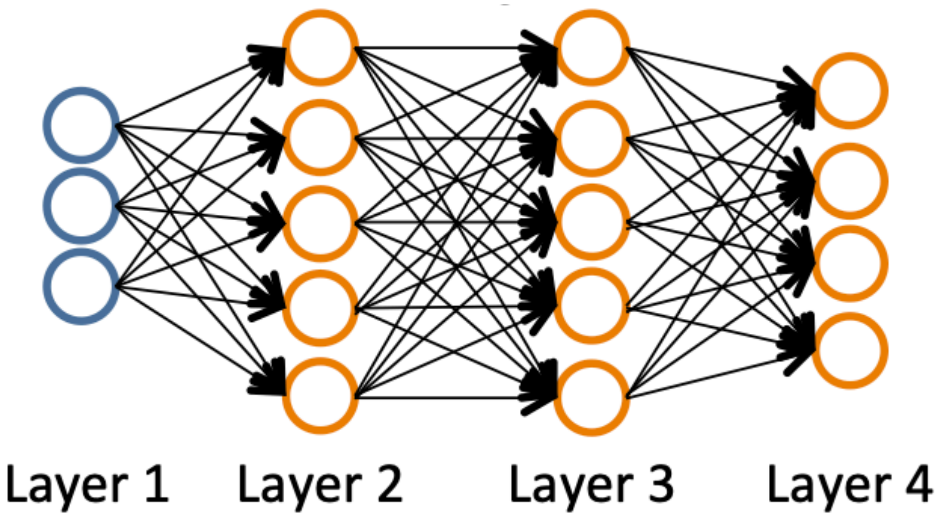
- You can classify handwritten digits with just a few (<20) lines of code, without knowing much about mathematics behind
- Classify text and extract deep insights from it (Sentiment Analysis)
- Deep learning is very iterative process so you can implement your idea within 30-40 minutes and improve upon it
  - Code less think more
- And many more just by using concepts from this class

Projects Which we cover today

- Digit Recognition (Using multilayer perceptron)
- Image Classification (Using Convolutional Neural Networks)
- Lean how you can classify a image in real time taken from your mobile phone (Using ResNet-50)
- All these in just 1 class 😃

Neural Network

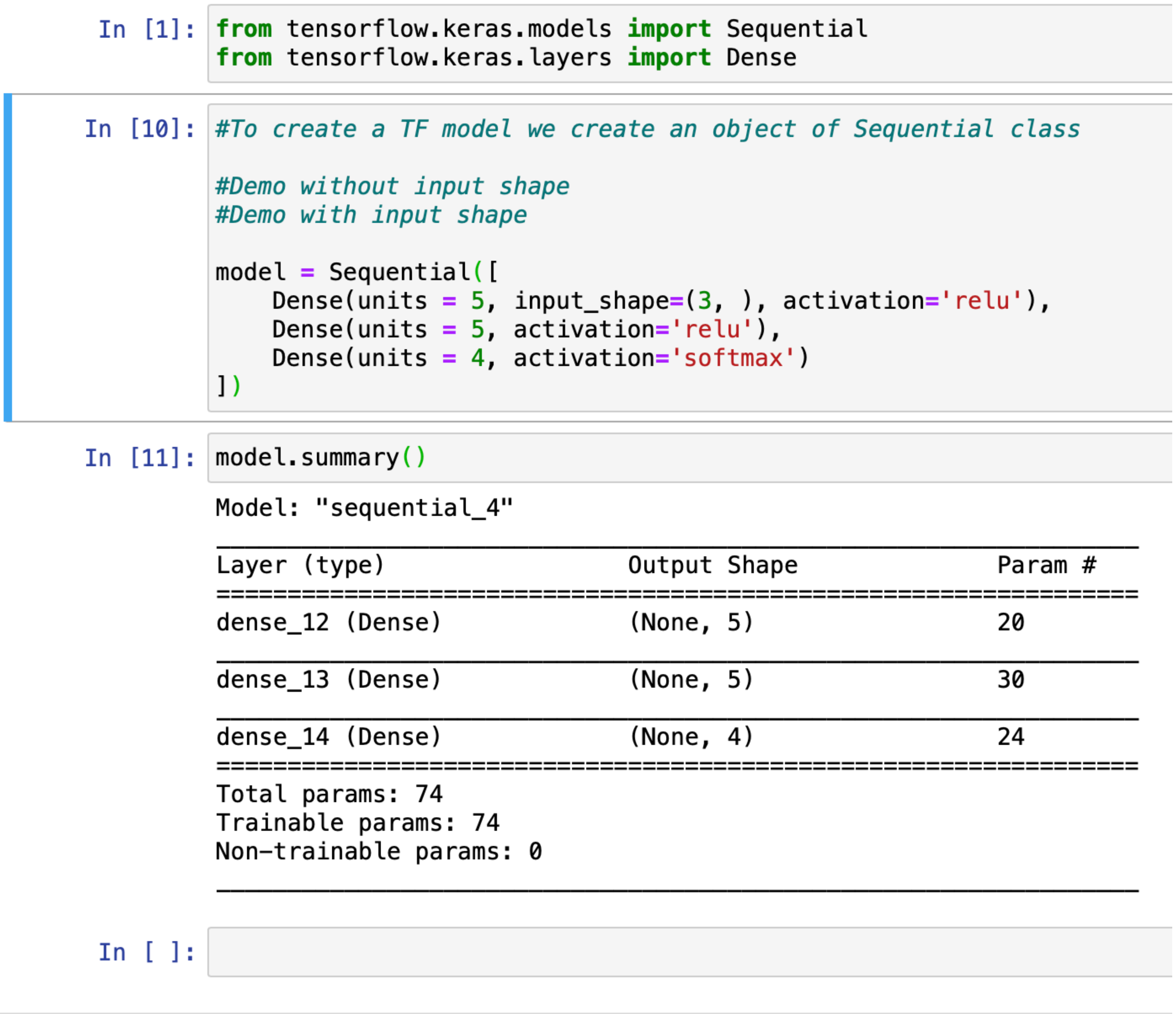
- Think of them as function generators (Explain how)
- We know how NN looks like (Image)



- Input shape depends on dataset
- Output shape depends on task (Digit classification eg)
- In between we can put any complex structure
- Once you have the NN image which you want to draw, lets see how to put that in code…

Multilayer Perceptron in TF 2.0

Image to code: Demo_1



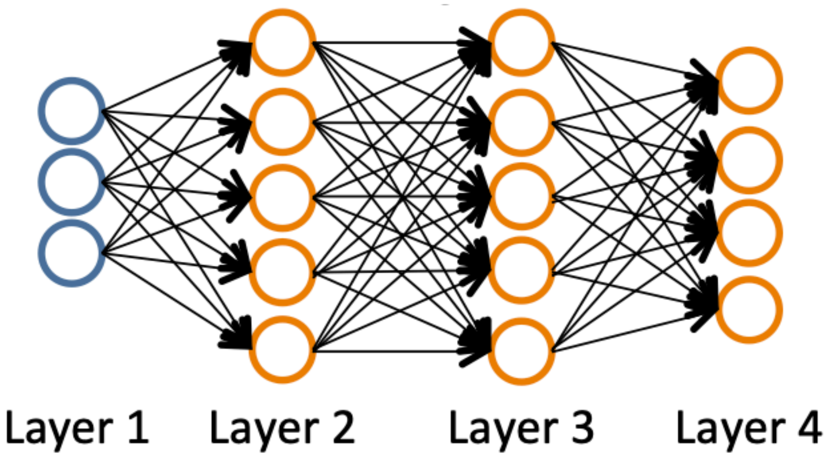
```
In [1]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense
```

```
In [10]: #To create a TF model we create an object of Sequential class

         #Demo without input shape
         #Demo with input shape

         model = Sequential([
             Dense(units = 5, input_shape=(3, ), activation='relu'),
             Dense(units = 5, activation='relu'),
             Dense(units = 4, activation='softmax')
         ])
```

```
In [11]: model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_12 (Dense)             (None, 5)                 20        
_________________________________________________________________
dense_13 (Dense)             (None, 5)                 30        
_________________________________________________________________
dense_14 (Dense)             (None, 4)                 24        
=================================================================
Total params: 74
Trainable params: 74
Non-trainable params: 0
_________________________________________________________________
```

```
In [ ]:
```

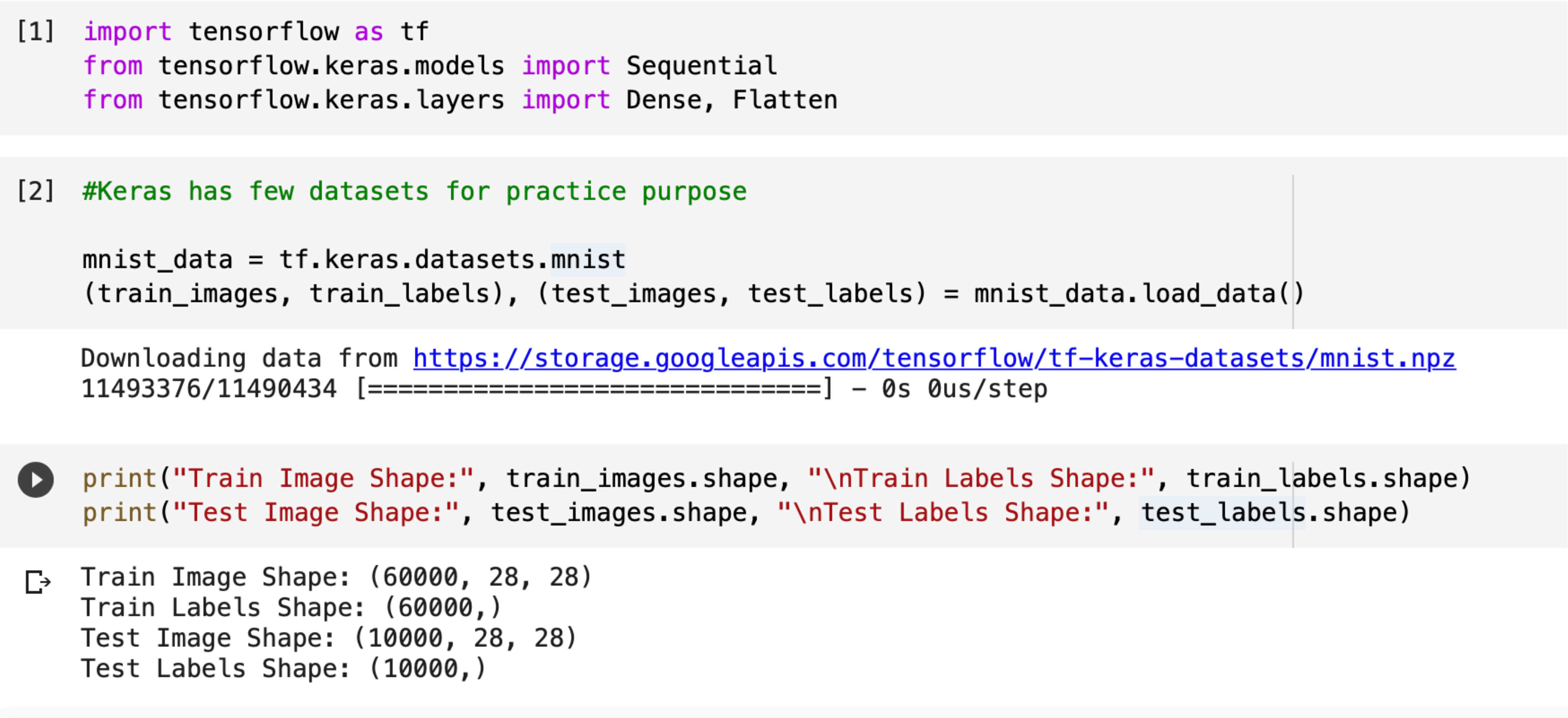


Terminologies

- **Activation:** Used to introduce non linearity in NN’s
    - Final layer “Sigmoid” for classification
    - Intermediate Layers “Relu” (Typically, but not always)
- **Loss Function:** Quantifies the amount by which predicted value differ from actual value. (**sparse_categorical_crossentropy** if final layer has **Sigmoid**)
- **Optimiser:** Gradient descend, Adam etc…
- **Metrics:** AUC, accuracy, False negatives etc etc…

Digit Classification

Lets classify some handwritten digits



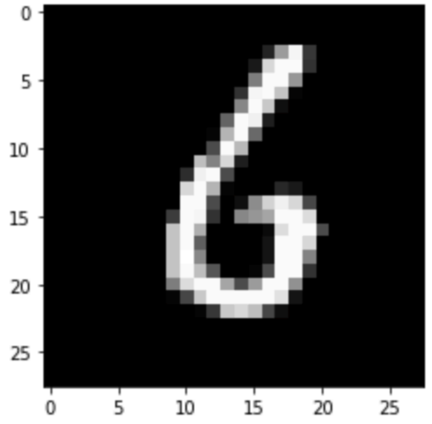
```
[1]  import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Flatten
```

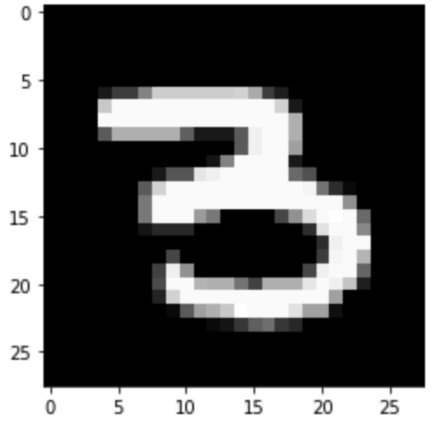
```
[2]  #Keras has few datasets for practice purpose

     mnist_data = tf.keras.datasets.mnist
     (train_images, train_labels), (test_images, test_labels) = mnist_data.load_data()
```

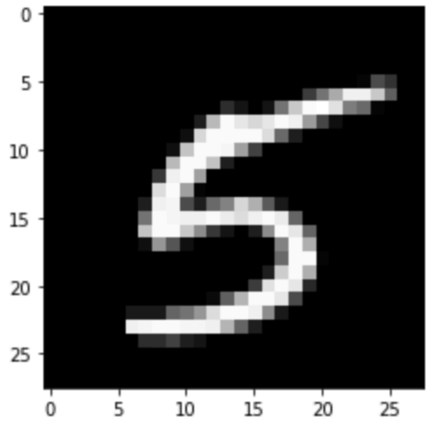
```
     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
     11493376/11490434 [==============================] - 0s 0us/step
```

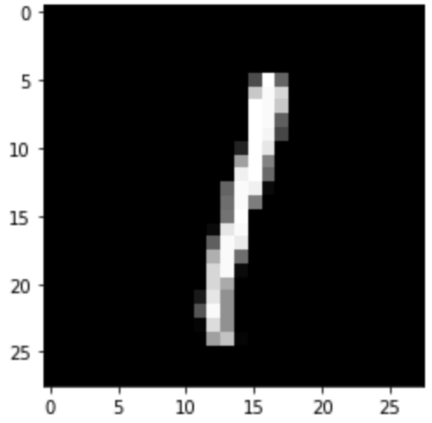
```
     print("Train Image Shape:", train_images.shape, "\nTrain Labels Shape:", train_labels.shape)
     print("Test Image Shape:", test_images.shape, "\nTest Labels Shape:", test_labels.shape)
```

```
     Train Image Shape: (60000, 28, 28)
     Train Labels Shape: (60000,)
     Test Image Shape: (10000, 28, 28)
     Test Labels Shape: (10000,)
```









Digit Recognition

Demo_2

- Typical Model Creation
    - Create Model
    - model.compile(…)
    - model.fit(…)
    - model.predict(…)

- What is Flatten() layer ?

Fashion Dataset

Demo_3

- Give idea about the dataset
- To gain better understanding I encourage you to play around with demo presented in class and
- Build your own classifier for MNSIT fashion dataset

Digit Recognition

Using CNN. Demo_4

- Let’s increase the accuracy of our model to >95% by using CNN
- Show the difference between number of parameters and accuracy of CNN and MLP
- CNN is proven to be good for images
- CNN
    - Convolution Layers (Filter, Stride, Paddings)
    - Pooling Layers (Max and Min)

- You should be able to build your own NN models
- Able to understand code written in TF
- Train your models
- Test the performance of your models
- We have covered 5-10% of what TF offers which you will be using 90% of the times

Train, Test & Validation Split

Demo_5

- Train set is to train your model
- Test set is to test the performance
- Why can’t you test using training data ?
- What is validation data ?
- How to give validation data to TF ?
    - Simple just pass an extra argument to model.fit(…)
- Uses of validation data
    - Helps tune the hyper-parameters of the model
    - One of the ways to detect overfitting

Overfitting & How to Identify it

Demo_6

- Overfitting is condition when your model gives GREAT accuracy for training data but fails to generalise for UNSEEN data
- How to identify: Demo_ (Plot between train and test loss)
- How to rectify ?
    - Get more data
    - Modify the model
    - Use regularisation techniques

Regularisation Techniques in TF

- Dropout Layer (Usage: “Dropout(rate)”)
    - Interesting, we are destroying few connections
    - It has the effect that each weight connection between 2 layers is set to 0 with probability ‘rate’
- Other Methods
    - Bath Normalisation Layer
    - Callbacks (Very Interesting and powerful)

Callbacks

Cover what it can do

- We have ability to monitor performance of model, callbacks gives us ability to perform some action based on the performance
    - EarlyStopping(): Stop training if performance of model is not increasing for certain epochs (Demo_6)
    - There are many more
- We can use callbacks already available or define our own callbacks
- See documentation…

Saving Models

Demo_7

- Large NN's trainings can take weeks
- We want to save models to have a backup in case system stops
- Ability to share model
- Ability to train model in Server and use on mobile
- What can be save:
    - Save weights only
    - Save complete model
- How to save:
    - Manually using model.save()
    - Using callbacks (Save automatically at the end of every epoch)
    - Save just one model which has given best performance
- Native TF format vs Keras format (.h5, saved in hdf5 format)

Keras Applications

- In Keras pre-trained models are referred to as Applications
- https://keras.io/api/applications/
- Explain include top argument

Classification Demo with ResNet-50

Demo_8

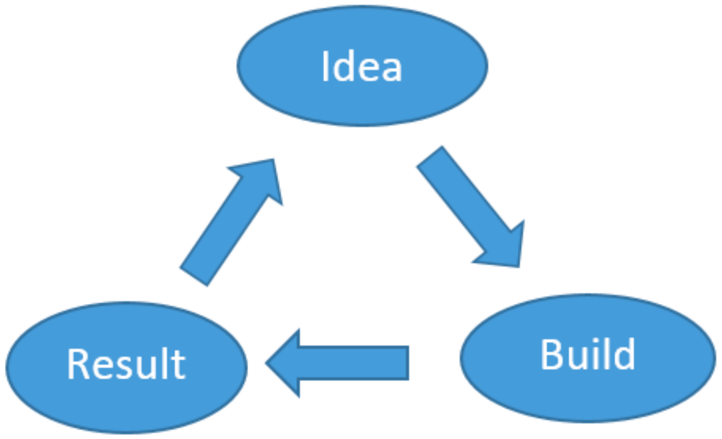
- How to load pre trained model using Keras API
- Load ResNet-50 model
- See model info
- Show real time image classification

Tensorflow Hub

- Visit TF Hub page at: https://www.tensorflow.org/hub
- Its a separate library and needs to be installed using commands…
- Explore on your own

Conclusion

- You can do a lot of ML even without knowing ML
- Use the knowledge of TF framework to quickly build and test your model
- As ML specially DL is highly iterative process
- These kind of frameworks if used correctly simplifies Build and Result phase so that you can spend much time on **Ideation**

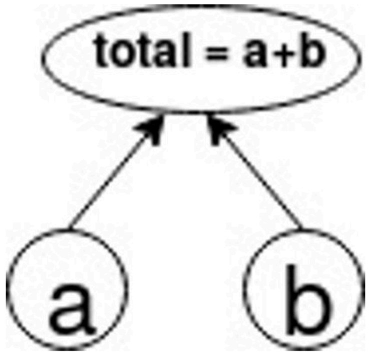


TF 1.0

TF as mathematical library

- A tensorflow program has two phases:
  - Define a graph (series of operations)
  - Execute the operations in the graph
- Define a graph

```
a = tf.constant(3.0, dtype=tf.float32)
b = tf.constant(4.0, dtype=tf.float32)
total = a + b
```



- Execute the graph

```
sess = tf.Session()
result = sess.run(total)
```

Multilayer Perceptron in TF 1.0

- Demo

Thank-you