Electronics & ICT Academy
National Institute of Technology, Warangal

Post Graduate Program in
Artificial Inteliigence & Machine Learning

Deep Learning
Question Bank

edureka!

Deep Learning

Table of Content

Module-2: Introduction to TensorFlow 2.0

Google's Open Source Machine Learning Framework created for machine learning tasks. It is a comphrensive and flexible resource backed by a vast community and libraries for easy build and deployment of ML powered applications. In this question bank we will be learning how to use tensorflow in multiple scenarios.

In [ ]:
```
!pip install tensorflow-gpu==2.0.0
```

Scenario-1: Fashion MNIST

Zalando's article images with a dimension of 28x28 which are grayscale. The images are associated with 10 different classes. The dataset was intended to replace the original MNIST dataset which contained the images of the digits.

Problem Statement

The aim is to classifiy the images in mulitple categories.

Dataset Description

- **Label**: *Class*
- **0**: *T-shirt/top*
- **1**: *Trouser*
- **2**: *Pullover*
- **3**: *Dress*
- **4**: *Coat*
- **5**: *Sandal*
- **6**: *Shirt*
- **7**: *Sneaker*
- **8**: *Bag*
- **9**: *Ankle boot*

Tasks to be Performed:

- Read the dataset and perform exploratory data analysis over the dataset. **Beginner**
- Build a sequential model. **Easy**
- Optimize the model using adam optimizer and Cross Entropy as loss function. **Intermediate**
- Evaluate the model based on the accuracy. **Intermediate**
- Plot the predictions of the model against the orginal test image. **Advanced**

Topics Covered:

- Sequential Model
- Adam Optimizer

Question-1: Read the dataset and perform exploratory data analysis over the dataset.

In [ ]:
```
# Import required libraries
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprec
ated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

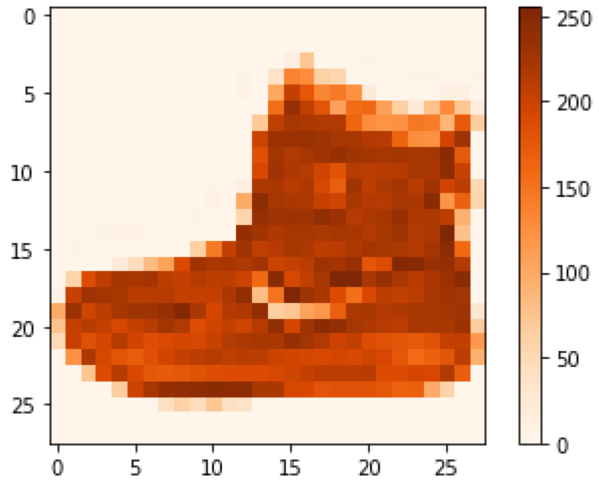
In [ ]:
```
# We are using dataset present in the tensorflow library. The function return two tuples in form of (x,y),(a,b) which
# can be interpreted as training set(image,label) and testing set(image, label).
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

In [ ]:
```
print('The shape of the train_data: ',train_images.shape)
print('The shape of the test_data: ',test_images.shape)
```

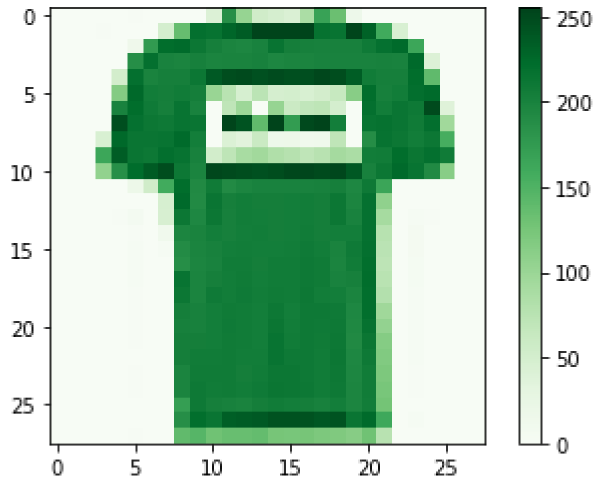
```
The shape of the train_data:  (60000, 28, 28)
The shape of the test_data:  (10000, 28, 28)
```

Let's try visualizing some of the images

In [ ]:
```
plt.figure()
plt.imshow(train_images[0],cmap='Oranges')
plt.colorbar()
plt.grid(False)
plt.show()
```



In [ ]:
```
plt.figure()
plt.imshow(train_images[1],cmap='Greens')
plt.colorbar()
plt.grid(False)
plt.show()
```



In [ ]:
```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

In [ ]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap='Blues')
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

We have visualized the images in a subplot.

Question-2: Build a sequential model.

In [ ]:
```
# Building a sequential model where Flatten layer convert 28x28 grid image into 784 single dimension.
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])
```

tf.keras.layers.Flatten transforms a 2D image to 1-dimensional(28 * 28 = 784 pixels).

Question-3: Optimize the model using adam optimizer and Cross Entropy as loss function.

In [ ]:
```
model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accura
cy'])
```

In [ ]:
```
model.fit(train_images, train_labels, epochs=10)
```

```
Train on 60000 samples
Epoch 1/10
60000/60000 [==============================] - 6s 101us/sample - loss: 0.4994 - accuracy: 0.8257
Epoch 2/10
60000/60000 [==============================] - 5s 91us/sample - loss: 0.3755 - accuracy: 0.8651
Epoch 3/10
60000/60000 [==============================] - 6s 101us/sample - loss: 0.3374 - accuracy: 0.8771
Epoch 4/10
60000/60000 [==============================] - 6s 100us/sample - loss: 0.3118 - accuracy: 0.8856
Epoch 5/10
60000/60000 [==============================] - 5s 88us/sample - loss: 0.2946 - accuracy: 0.8924
Epoch 6/10
60000/60000 [==============================] - 5s 89us/sample - loss: 0.2806 - accuracy: 0.8963
Epoch 7/10
60000/60000 [==============================] - 5s 91us/sample - loss: 0.2668 - accuracy: 0.9001
Epoch 8/10
60000/60000 [==============================] - 6s 92us/sample - loss: 0.2584 - accuracy: 0.9027
Epoch 9/10
60000/60000 [==============================] - 5s 89us/sample - loss: 0.2480 - accuracy: 0.9063
Epoch 10/10
60000/60000 [==============================] - 5s 90us/sample - loss: 0.2417 - accuracy: 0.9100
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f342830ce48>
```

Question-4: Evaluate the model based on the accuracy.

In [ ]:
```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
10000/1 - 1s - loss: 0.2471 - accuracy: 0.8838

Test accuracy: 0.8838
```

Question-5: Plot the predictions of the model against the orginal test image.

In [ ]:
```
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
```

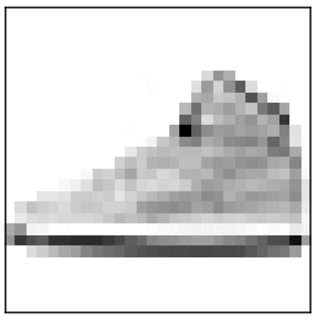
In [ ]:
```
predictions = probability_model.predict(test_images)
```

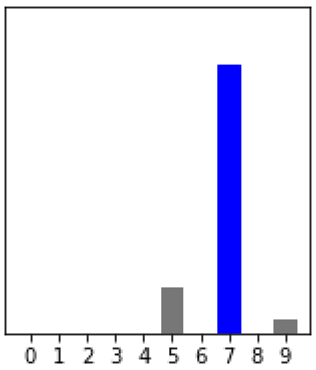
In [ ]:
```
np.argmax(predictions[0])
```

Out[ ]:
```
9
```

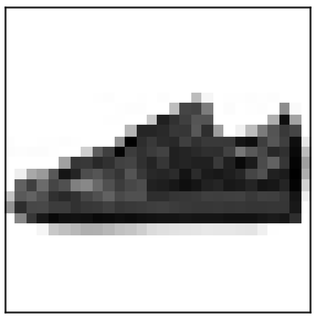
In [ ]:
```
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array, true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array, true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

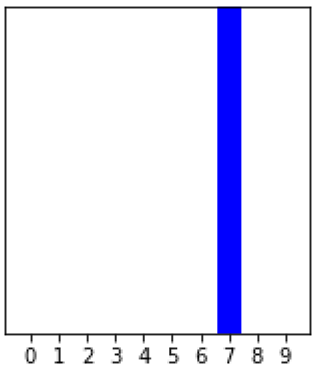
In [ ]:
```
i = 45
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```


Sneaker 82% (Sneaker)

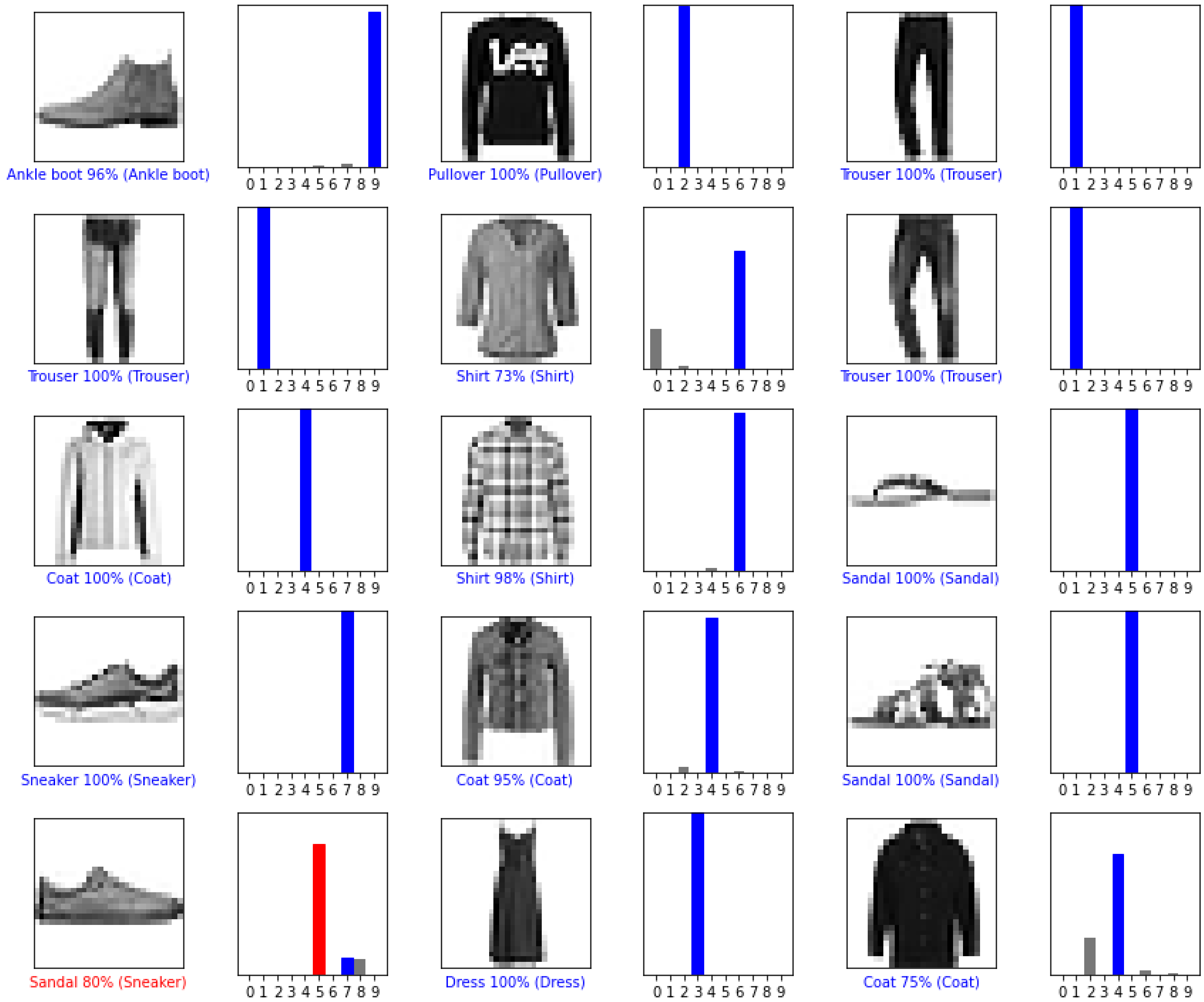


In [ ]:
```
i = 22
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```


Sneaker 100% (Sneaker)



In [ ]:
```
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```



We have create a grid layout where we can see how our model is able to classify the images.

Scenario-2: Fuel Efficiency

The dataset contains the specifications of a number of cars along with the fuel efficiency of each car. The aim is to create a model that can predict the efficiency of a car based on the details provided.

Dataset Decription:

- **mpg**: continuous, Miles Per Galon
- **cylinders**: multi-valued, discrete, Number of cylinders
- **displacement**: continuous
- **horsepower**: continuous
- **weight**: continuous
- **acceleration**: continuous
- **model year**: multi-valued, discrete
- **origin**: multi-valued, discrete
- **car name**: string (unique for each instance)

Tasks to be Performed:

- Read the dataset using Kaggle API and process the missing values. **Beginner**
- Perform EDA over the data and normalize the dataset. **Intermediate**
- Build a Sequential model using dense layers with relu as activation function and RMSprop as optimizer. **Intermediate**
- Fit the model using EpochDots as callback function from tensorflow docs. **Intermediate**
- Plot the history of the model using HistoryPlotter for mean absolute error and mean squared error. **Advanced**

Topics Covered:

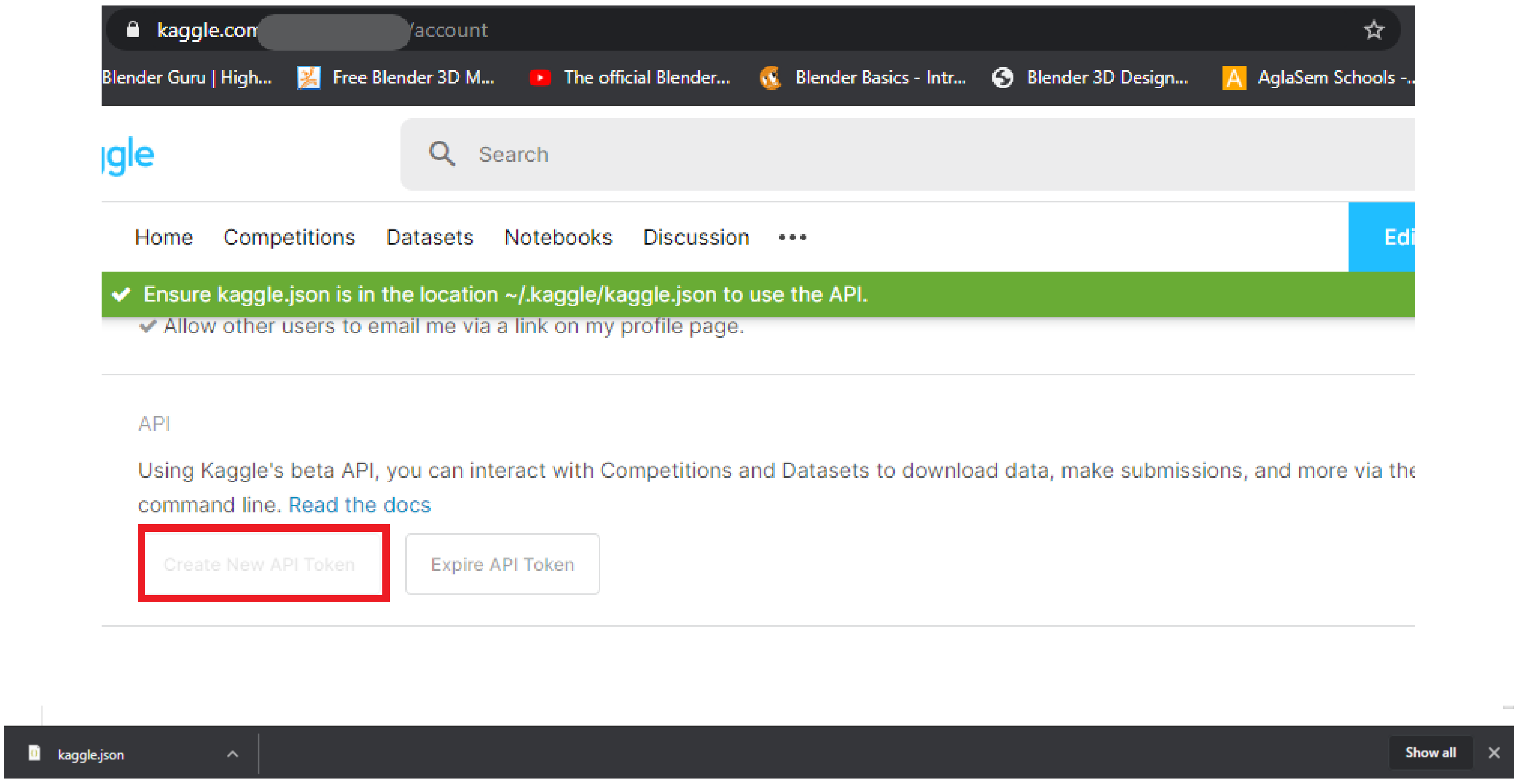
- Sequential Model
- RMSprop

Question-1: Read the dataset using Kaggle API and process the missing values.

- Easy way to import data from kaggle

  Kaggle is world's largest data science community. Kaggle provides a number of tools and resources for free. But downloading and uploading the large datasets can be a hassle and tiring. Instead, we can make use of kaggle APIs to fetch the datasets.

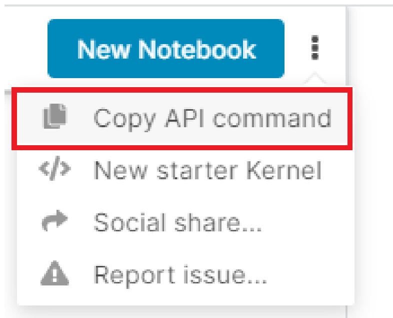
- Go to **My Account** and click on **Create New API Token**.
- A file named **kaggle.json** will get downloaded containing your **username** and **token key**.



- Create a folder named **kaggle** on drive where you will store all the kaggle datasets.
- Upload your **kaggle.json** file into the respective folder
- Mount the drive using the below code:

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

- Go to kaggle and copy the API Command to download the dataset.



In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee649
1hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%
20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2
f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/gdrive
```

In [ ]:
```
# Changing the working directory

%cd /content/gdrive/My Drive

# Create an environment variable for kaggle config directory
import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content/gdrive/My Drive/datasets"
```

```
/content/gdrive/My Drive
```

In [ ]:
```
!kaggle datasets download -d uciml/autompg-dataset
```

```
Downloading autompg-dataset.zip to /content/gdrive/My Drive
  0% 0.00/6.31k [00:00<?, ?B/s]
100% 6.31k/6.31k [00:00<00:00, 862kB/s]
```

In [ ]:
```
# Unzipping the zip files and deleting the zip files
!unzip \autompg-dataset.zip  && rm autompg-dataset.zip
```

```
Archive:  autompg-dataset.zip
  inflating: auto-mpg.csv
```

In [ ]:
```
import pandas as pd
column_names = ['MPG','Cylinders','Displacement','Horsepower','Weight',
                'Acceleration', 'Model Year', 'Origin']
raw_dataset = pd.read_csv('auto-mpg.csv', names=column_names,
                      na_values = "?",skiprows=1)

data = raw_dataset.copy()
data.head()
```

Out[ ]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320
18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite
16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst
17.0	8	302.0	140.0	3449	10.5	70	1	ford torino

In [ ]:
```
data.drop('Origin',axis=1,inplace=True)
```

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?': [any(data[x].isnull()) for x in data.columns],
                 'Count_':[sum(data[y].isnull()) for y in data.columns]})
```

In [ ]:
```
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_
2	Displacement	True	6
0	MPG	False	0
1	Cylinders	False	0
3	Horsepower	False	0
4	Weight	False	0
5	Acceleration	False	0
6	Model Year	False	0

In [ ]:
```
data.Displacement.fillna(data.Displacement.mean(),inplace=True)
```

In [ ]:
```
data.dtypes
```

Out[ ]:
```
MPG               int64
Cylinders       float64
Displacement    float64
Horsepower        int64
Weight          float64
Acceleration      int64
Model Year        int64
dtype: object
```

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?': [any(data[x].isnull()) for x in data.columns],
                 'Count_':[sum(data[y].isnull()) for y in data.columns]})
```

In [ ]:
```
miss.sort_values(by='Count_',ascending=False)
```

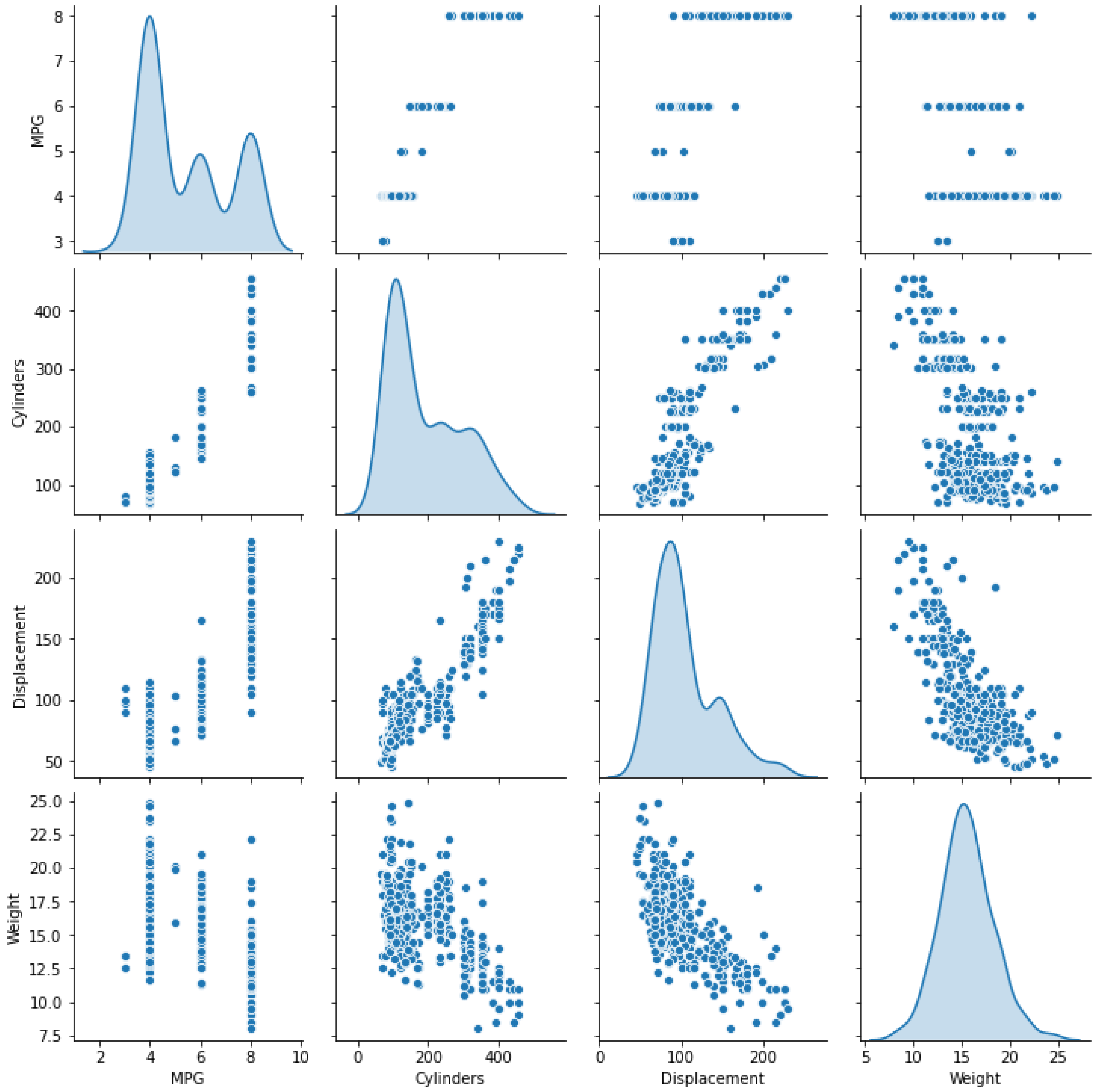
Out[ ]:

	Col_name	Missing value?	Count_
0	MPG	False	0
1	Cylinders	False	0
2	Displacement	False	0
3	Horsepower	False	0
4	Weight	False	0
5	Acceleration	False	0
6	Model Year	False	0

Question-2: Perform EDA over the data and normalize the dataset.

In [ ]:
```
# Explore the data
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(data[["MPG", "Cylinders", "Displacement", "Weight"]], diag_kind="kde")
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x7ffa0b80d208>



In [ ]:
```
stats = data.describe()
stats.pop("MPG")
stats = stats.transpose()
stats
```

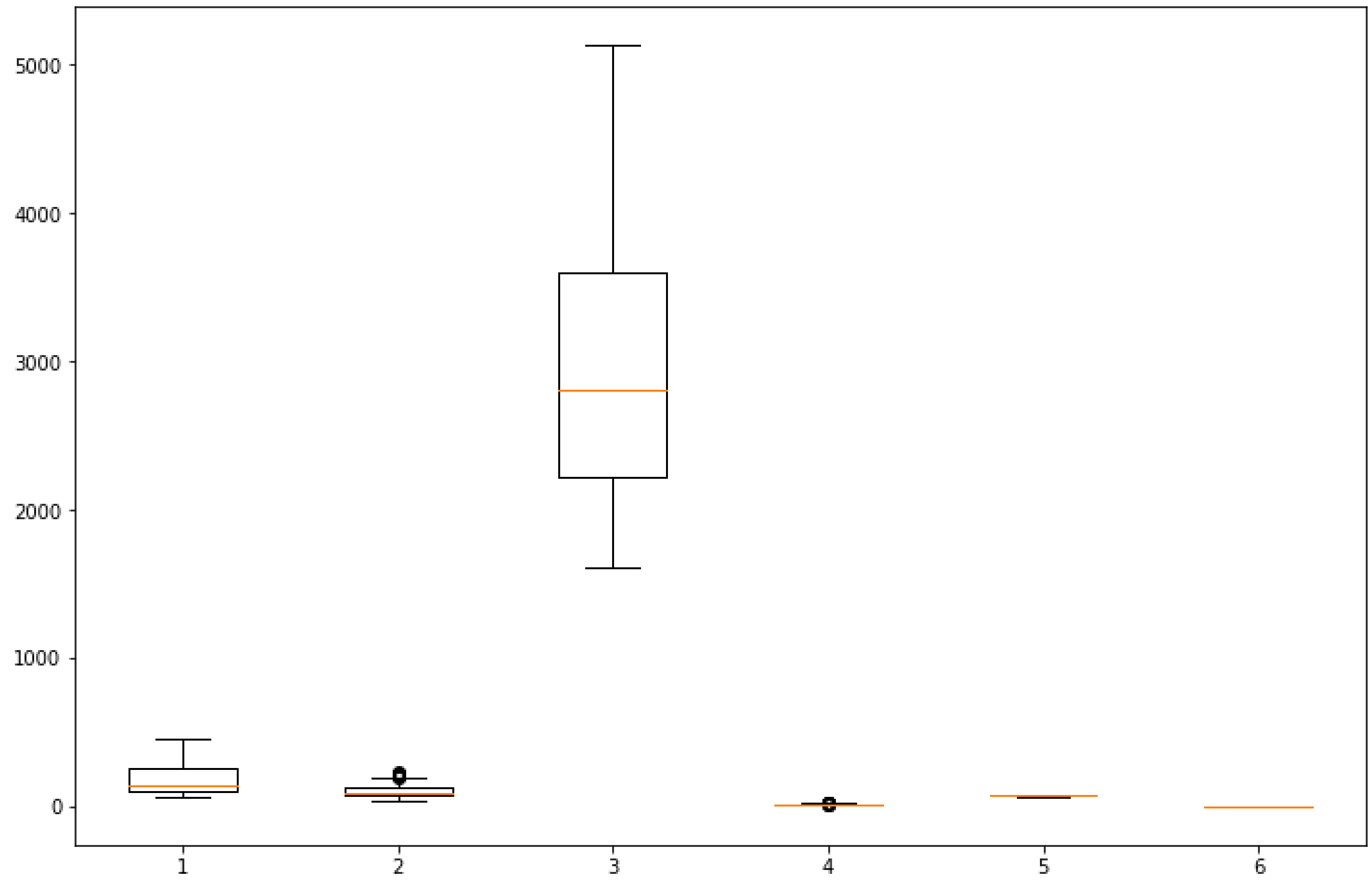
Out[ ]:

	count	mean	std	min	25%	50%	75%	max
Cylinders	398.0	193.425879	104.269838	68.0	104.250	148.5	262.000	455.0
Displacement	398.0	104.469388	38.199187	46.0	76.000	95.0	125.000	230.0
Horsepower	398.0	2970.424623	846.841774	1613.0	2223.750	2803.5	3608.000	5140.0
Weight	398.0	15.568090	2.757689	8.0	13.825	15.5	17.175	24.8
Acceleration	398.0	76.010050	3.697627	70.0	73.000	76.0	79.000	82.0
Model Year	398.0	1.572864	0.802055	1.0	1.000	1.0	2.000	3.0

In [ ]:
```
from sklearn.model_selection import train_test_split
```

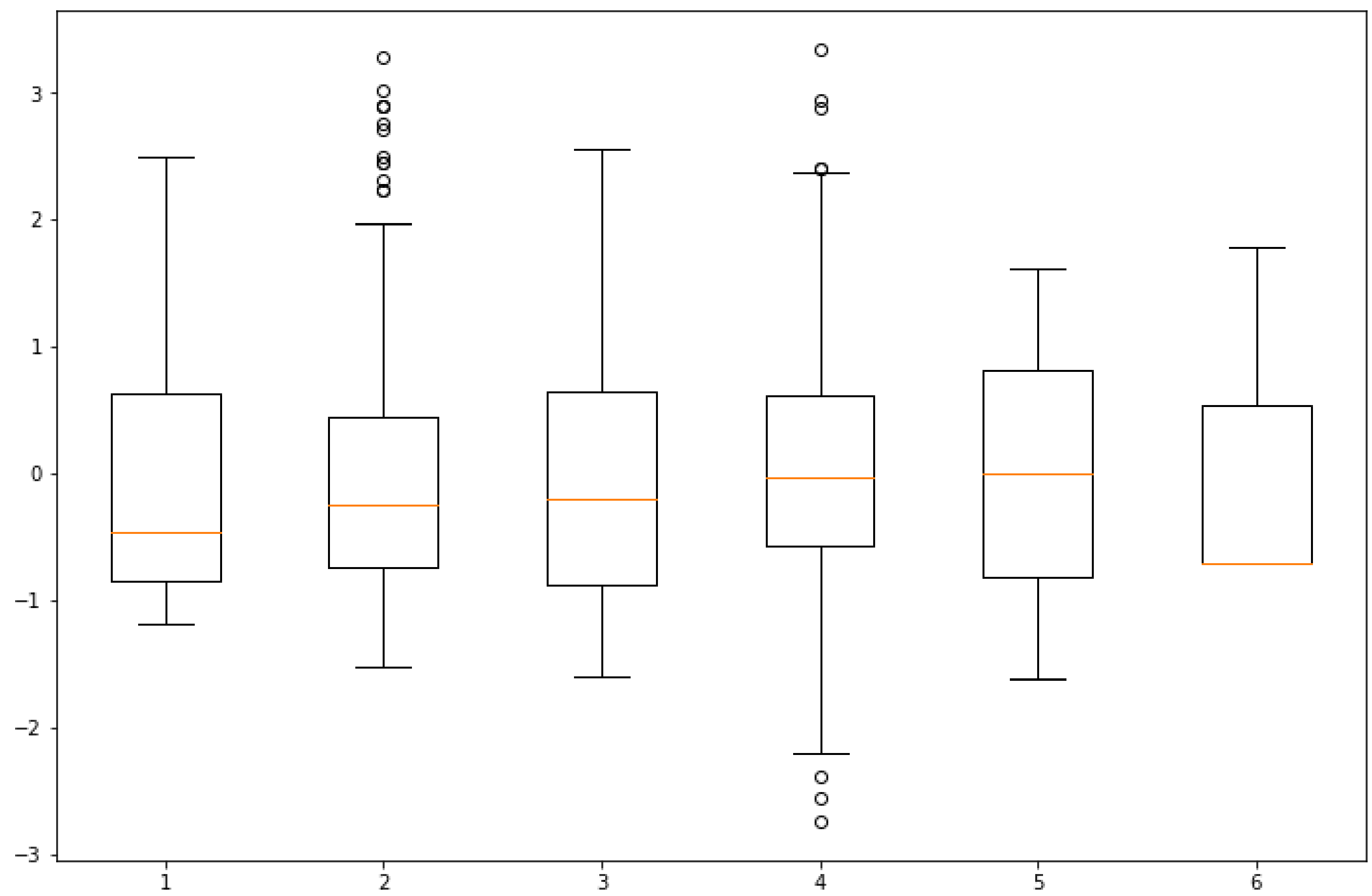
In [ ]:
```
X=data.drop('MPG',axis=1)
y=data.MPG
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,test_size=0.3, random_state=101)
```

In [ ]:
```
# boxplot
plt.boxplot(data[stats.index].T)
plt.show()
```



In [ ]:
```
def norm(x):
  return (x - stats['mean']) / stats['std']
normed_train_data = norm(X_train)
normed_test_data = norm(X_test)
```

In [ ]:
```
plt.boxplot(normed_train_data[stats.index].T)
plt.show()
```



Question-3: Build a Sequential model using dense layers with relu as activation function and RMSprop as optimizer

In [ ]:
```python
# Building the model
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
def build_model():
  model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=[len(X_train.keys())]),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model
```

In [ ]:
```python
model = build_model()
```

In [ ]:
```python
# we have 3 layers where first 2 layers are dense layer that return 64 outputs while last layer is output layer
# with 1 output as value
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 64)                448
_________________________________________________________________
dense_13 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 65
=================================================================
Total params: 4,673
Trainable params: 4,673
Non-trainable params: 0
_________________________________________________________________
```

Question-4: Fit the model using EpochDots as callback function from tensorflow docs.

In [ ]:
```python
!pip install git+https://github.com/tensorflow/docs
```

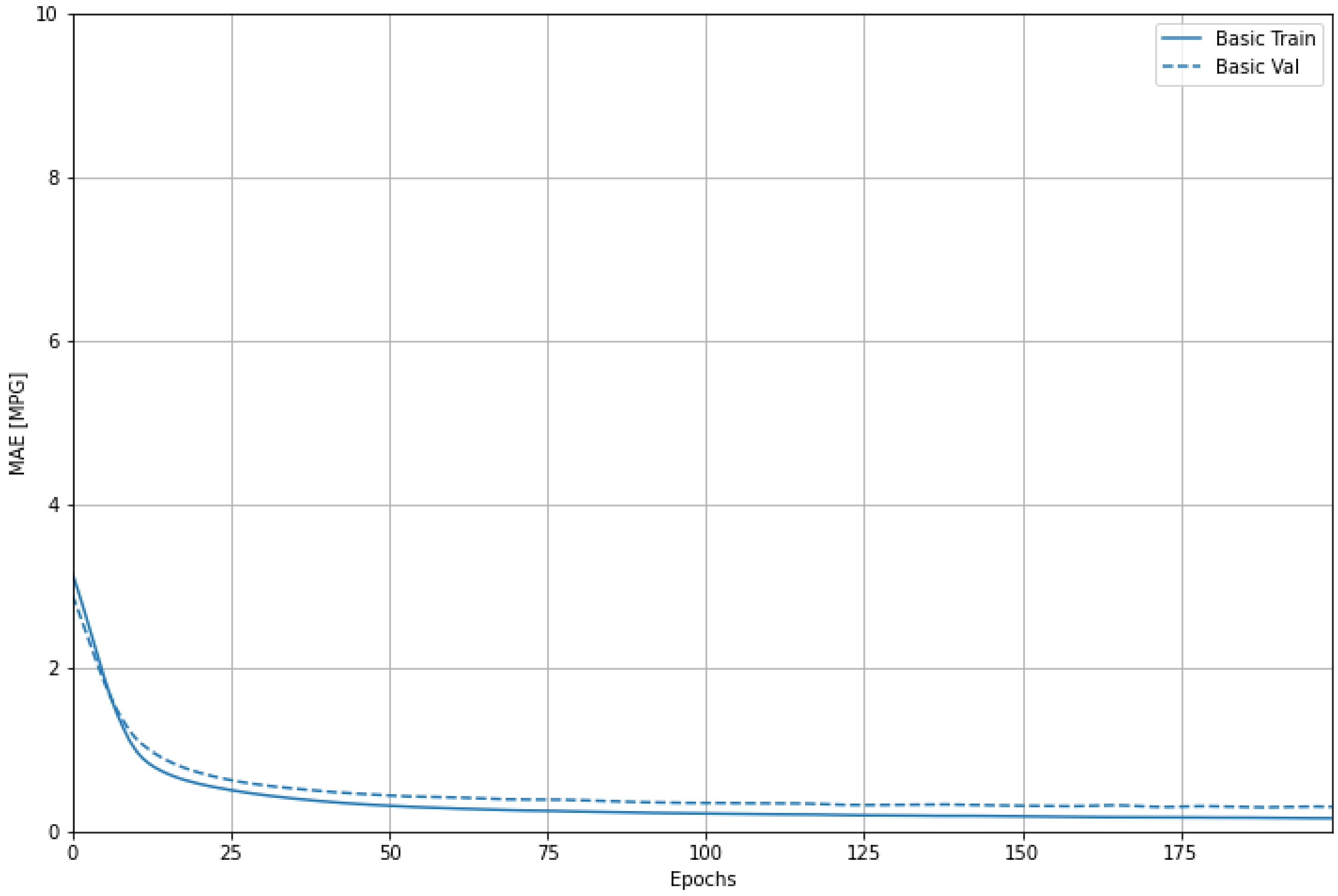
```
Collecting git+https://github.com/tensorflow/docs
  Cloning https://github.com/tensorflow/docs to /tmp/pip-req-build-dobpjcl6
  Running command git clone -q https://github.com/tensorflow/docs /tmp/pip-req-build-dobpjcl6
Requirement already satisfied (use --upgrade to upgrade): tensorflow-docs===0.0.0f82f2a94252f97efd2cdbc57408469e1d4cd
9774- from git+https://github.com/tensorflow/docs in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: astor in /usr/local/lib/python3.6/dist-packages (from tensorflow-docs===0.0.0f82f2a942
52f97efd2cdbc57408469e1d4cd9774-) (0.8.1)
Requirement already satisfied: absl-py in /usr/local/lib/python3.6/dist-packages (from tensorflow-docs===0.0.0f82f2a9
4252f97efd2cdbc57408469e1d4cd9774-) (0.9.0)
Requirement already satisfied: protobuf in /usr/local/lib/python3.6/dist-packages (from tensorflow-docs===0.0.0f82f2a
94252f97efd2cdbc57408469e1d4cd9774-) (3.10.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages (from tensorflow-docs===0.0.0f82f2a94
252f97efd2cdbc57408469e1d4cd9774-) (3.13)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from absl-py->tensorflow-docs===0.0.0f8
2f2a94252f97efd2cdbc57408469e1d4cd9774-) (1.12.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from protobuf->tensorflow-docs==
=0.0.0f82f2a94252f97efd2cdbc57408469e1d4cd9774-) (47.3.1)
Building wheels for collected packages: tensorflow-docs
  Building wheel for tensorflow-docs (setup.py) ... done
  Created wheel for tensorflow-docs: filename=tensorflow_docs-0.0.0f82f2a94252f97efd2cdbc57408469e1d4cd9774_-cp36-non
e-any.whl size=119874 sha256=b30f0ee4007ee06cc05f3d79925065c6cfa8ef70204eeffeae975da6ec59fc7e
  Stored in directory: /tmp/pip-ephem-wheel-cache-hq5okrc8/wheels/eb/1b/35/fce87697be00d2fc63e0b4b395b0d9c7e391a10e98
d9a0d97f
Successfully built tensorflow-docs
```

In [ ]:
```
import tensorflow_docs as tfdocs
import tensorflow_docs.plots
import tensorflow_docs.modeling
EPOCHS = 200

history = model.fit(
  normed_train_data, y_train,
epochs=EPOCHS, validation_split = 0.2, verbose=0,callbacks=[tensorflow_docs.modeling.EpochDots(10)])
```

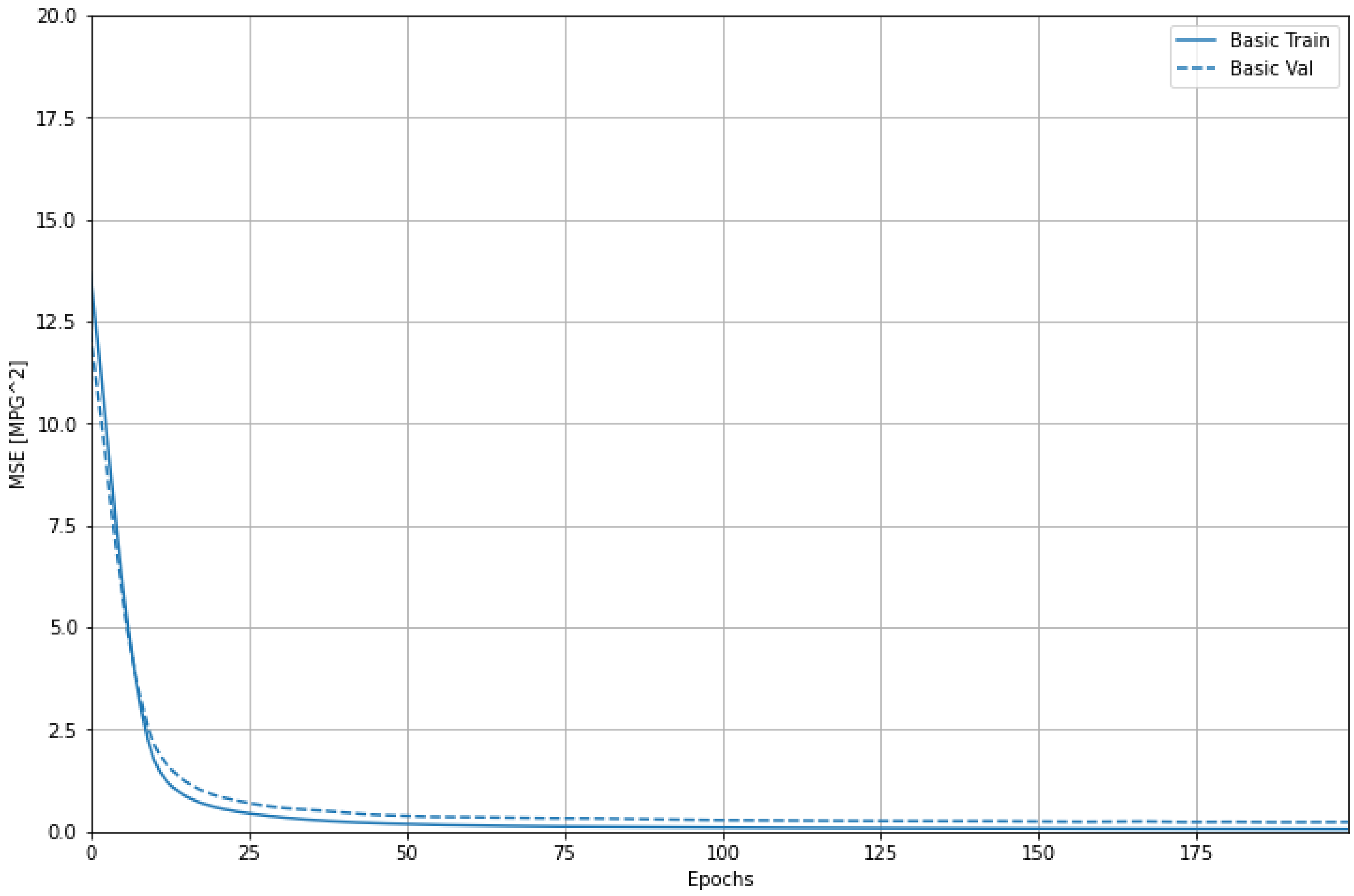
```
Epoch: 0, loss:27.4111,  mae:4.9569,  mse:27.4111,  val_loss:25.0129,  val_mae:4.6784,  val_mse:25.0129,
..........
Epoch: 10, loss:1.3973,  mae:0.8981,  mse:1.3973,  val_loss:1.9434,  val_mae:1.0988,  val_mse:1.9434,
..........
Epoch: 20, loss:0.5479,  mae:0.5620,  mse:0.5479,  val_loss:0.7897,  val_mae:0.6756,  val_mse:0.7897,
..........
Epoch: 30, loss:0.3358,  mae:0.4377,  mse:0.3358,  val_loss:0.5991,  val_mae:0.6064,  val_mse:0.5991,
..........
Epoch: 40, loss:0.2285,  mae:0.3623,  mse:0.2285,  val_loss:0.4327,  val_mae:0.4813,  val_mse:0.4327,
..........
Epoch: 50, loss:0.1988,  mae:0.3338,  mse:0.1988,  val_loss:0.3788,  val_mae:0.4489,  val_mse:0.3788,
..........
Epoch: 60, loss:0.1498,  mae:0.2807,  mse:0.1498,  val_loss:0.4005,  val_mae:0.4530,  val_mse:0.4005,
..........
Epoch: 70, loss:0.1188,  mae:0.2446,  mse:0.1188,  val_loss:0.3049,  val_mae:0.3746,  val_mse:0.3049,
..........
Epoch: 80, loss:0.1003,  mae:0.2189,  mse:0.1003,  val_loss:0.3000,  val_mae:0.3629,  val_mse:0.3000,
..........
Epoch: 90, loss:0.0870,  mae:0.2067,  mse:0.0870,  val_loss:0.2863,  val_mae:0.3375,  val_mse:0.2863,
..........
Epoch: 100, loss:0.0873,  mae:0.2065,  mse:0.0873,  val_loss:0.3015,  val_mae:0.3860,  val_mse:0.3015,
..........
Epoch: 110, loss:0.0712,  mae:0.1822,  mse:0.0712,  val_loss:0.2431,  val_mae:0.3076,  val_mse:0.2431,
..........
Epoch: 120, loss:0.0794,  mae:0.2035,  mse:0.0794,  val_loss:0.2475,  val_mae:0.3209,  val_mse:0.2475,
..........
Epoch: 130, loss:0.0829,  mae:0.2085,  mse:0.0829,  val_loss:0.2543,  val_mae:0.3285,  val_mse:0.2543,
..........
Epoch: 140, loss:0.0780,  mae:0.2004,  mse:0.0780,  val_loss:0.2440,  val_mae:0.3120,  val_mse:0.2440,
..........
Epoch: 150, loss:0.0602,  mae:0.1685,  mse:0.0602,  val_loss:0.2404,  val_mae:0.2996,  val_mse:0.2404,
..........
Epoch: 160, loss:0.0610,  mae:0.1737,  mse:0.0610,  val_loss:0.2395,  val_mae:0.3072,  val_mse:0.2395,
..........
Epoch: 170, loss:0.0774,  mae:0.2065,  mse:0.0774,  val_loss:0.2193,  val_mae:0.2903,  val_mse:0.2193,
..........
Epoch: 180, loss:0.0731,  mae:0.1947,  mse:0.0731,  val_loss:0.2147,  val_mae:0.2796,  val_mse:0.2147,
..........
Epoch: 190, loss:0.0501,  mae:0.1555,  mse:0.0501,  val_loss:0.2050,  val_mae:0.2646,  val_mse:0.2050,
..........
```

Question-5: Plot the history of the model using HistoryPlotter for mean absolute error and mean squared error.

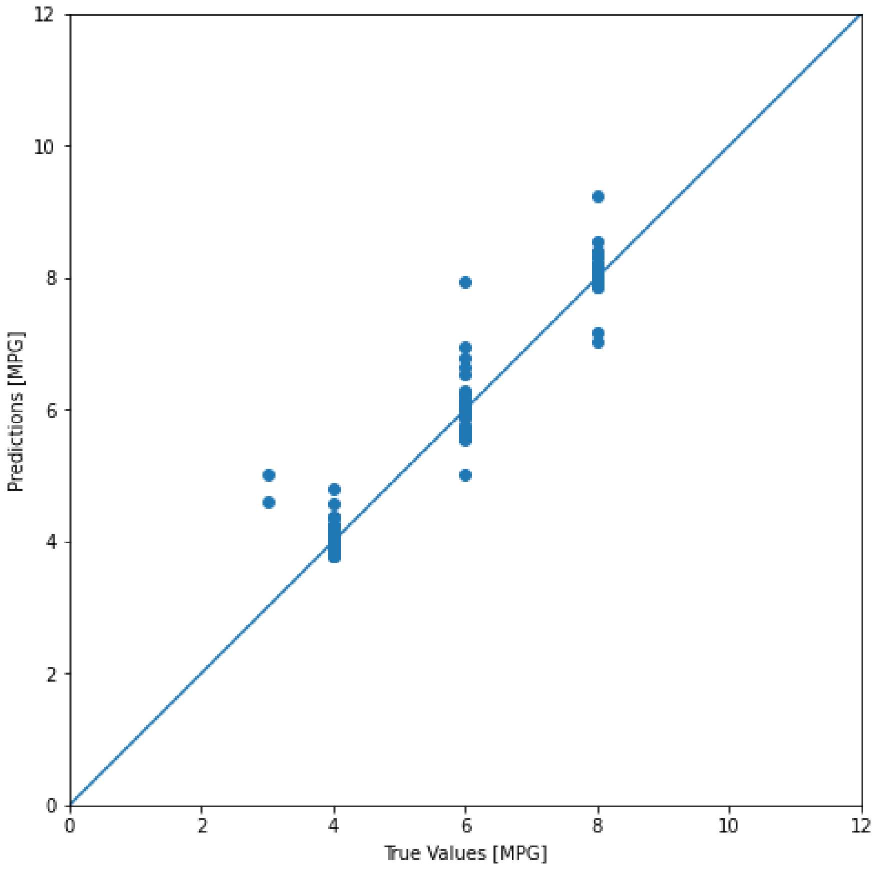
In [ ]:
```
plotter = tfdocs.plots.HistoryPlotter(smoothing_std=2)
plotter.plot({'Basic': history}, metric = "mae")
plt.ylim([0, 10])
plt.ylabel('MAE [MPG]')
plt.show()
```



In [ ]:
```
plotter.plot({'Basic': history}, metric = "mse")
plt.ylim([0, 20])
plt.ylabel('MSE [MPG^2]')
plt.show()
```



In [ ]:
```
test_predictions = model.predict(normed_test_data).flatten()

a = plt.axes(aspect='equal')
plt.scatter(y_test, test_predictions)
plt.xlabel('True Values [MPG]')
plt.ylabel('Predictions [MPG]')
lims = [0, 12]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```



Scenario-3: California Housing Data

The dataset if from 1990 California census data containing one row per census group. The dataset has various demographics and details captured. Based on this data we have to create a model that can determine the housing price of the house based on the details provided.

Dataset Description:

- **longitude**: A measure of how far west a house is; a higher value is farther west
- **latitude**: A measure of how far north a house is; a higher value is farther north
- **housingMedianAge**: Median age of a house within a block; a lower number is a newer building
- **totalRooms**: Total number of rooms within a block
- **totalBedrooms**: Total number of bedrooms within a block
- **population**: Total number of people residing within a block
- **households**: Total number of households, a group of people residing within a home unit, for a block
- **medianIncome**: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
- **medianHouseValue**: Median house value for households within a block (measured in US Dollars)

Tasks to be Performed:

- Read the dataset using Kaggle API and process the missing values. **Beginner**
- Perform EDA over the dataset. **Intermediate**
- Split the dataset into training and testing set. Create a sequential model using RMSprop optimizer. **Intermediate**
- Fit the model for using EpochDots and plot the history of the model using HistoryPlotter. **Adavanced**
- Plot a histogram of errors. **Intermediate**

Topics Covered:

- Sequential Model
- RMSprop

Question-1: Read the dataset using Kaggle API and process the missing values.

In [ ]:
```
!kaggle datasets download -d harrywang/housing
```

```
Downloading housing.zip to /content/gdrive/My Drive
  0% 0.00/400k [00:00<?, ?B/s]
100% 400k/400k [00:00<00:00, 27.2MB/s]
```

In [ ]:
```
!unzip \housing.zip  && rm housing.zip
```

```
Archive:  housing.zip
replace anscombe.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: anscombe.csv
  inflating: housing.csv
```

In [ ]:
```
import pandas as pd
data=pd.read_csv('housing.csv')
```

In [ ]:
```
data.head()
```

Out[ ]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_pr
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NE
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NE
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NE
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NE
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NE

In [ ]:
```
data.describe(include='all')
```

Out[ ]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_hou
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	206
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	2068
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	1153
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	149
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	1196
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	1797
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	2647
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	5000

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

In [ ]:
```
lb=LabelEncoder()
data.ocean_proximity=lb.fit_transform(data.ocean_proximity)
```

In [ ]:
```
data.head()
```

Out[ ]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_pr
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?': [any(data[x].isnull()) for x in data.columns],
                  'Count_':[sum(data[y].isnull()) for y in data.columns]})
```

In [ ]:
```
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

	Col_name	Missing value?	Count_
4	total_bedrooms	True	207
0	longitude	False	0
1	latitude	False	0
2	housing_median_age	False	0
3	total_rooms	False	0
5	population	False	0
6	households	False	0
7	median_income	False	0
8	median_house_value	False	0
9	ocean_proximity	False	0

In [ ]:
```
# Dropping null values
data.dropna(inplace=True)
```

In [ ]:
```
miss=pd.DataFrame({'Col_name':data.columns,'Missing value?': [any(data[x].isnull()) for x in data.columns],
                  'Count_':[sum(data[y].isnull()) for y in data.columns]})
```

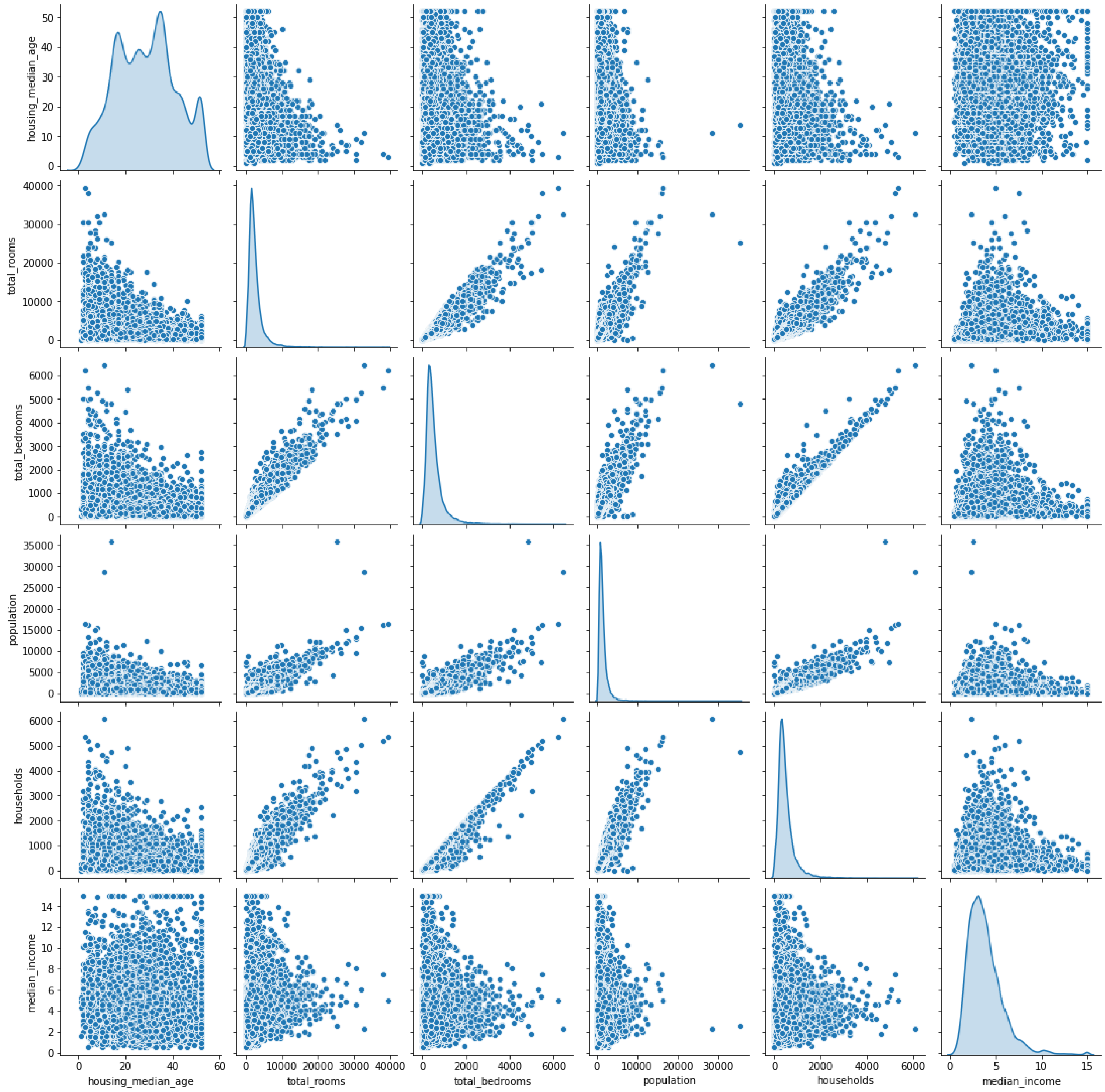
In [ ]:
```
miss.sort_values(by='Count_',ascending=False)
```

Out[ ]:

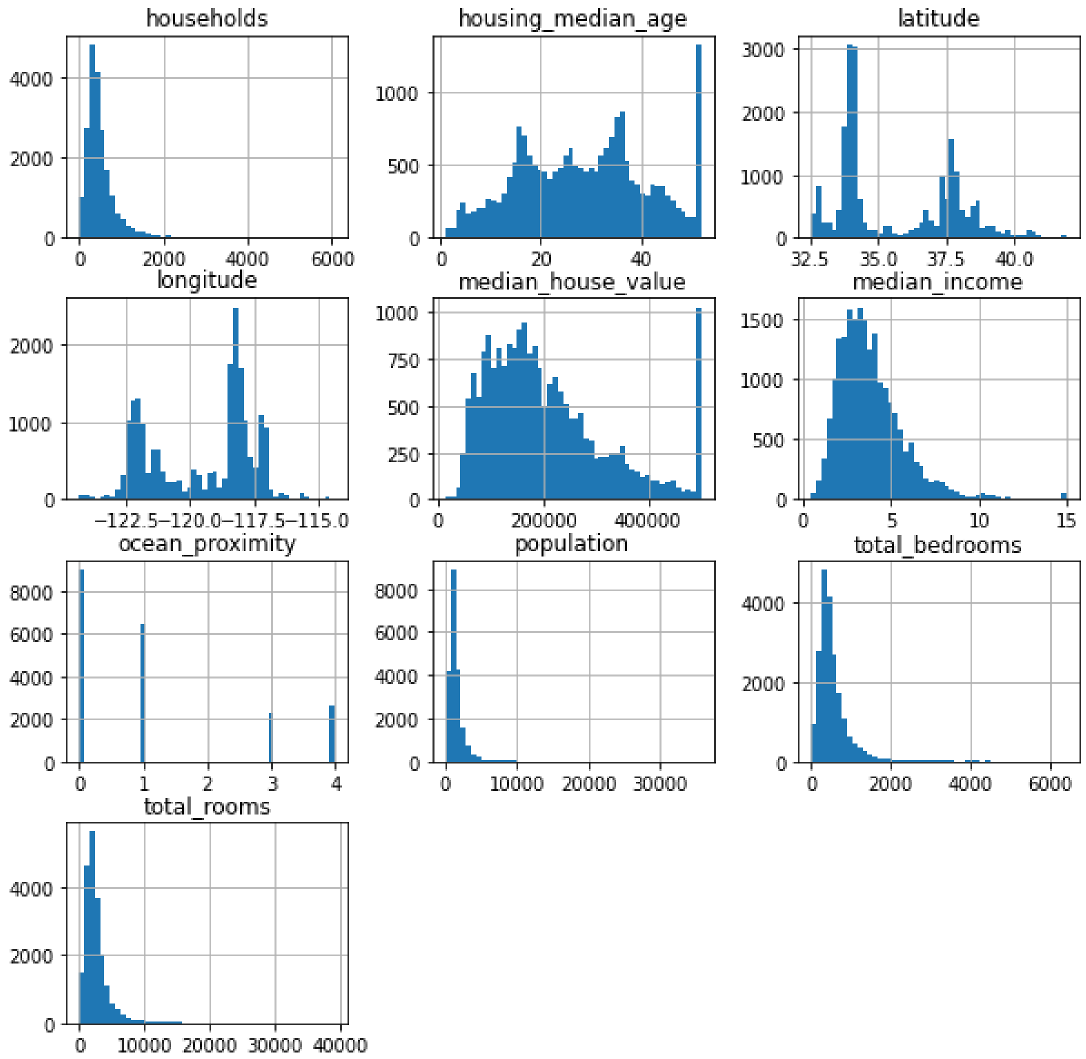
	Col_name	Missing value?	Count_
0	longitude	False	0
1	latitude	False	0
2	housing_median_age	False	0
3	total_rooms	False	0
4	total_bedrooms	False	0
5	population	False	0
6	households	False	0
7	median_income	False	0
8	median_house_value	False	0
9	ocean_proximity	False	0

Question-2: Perform EDA over the dataset.

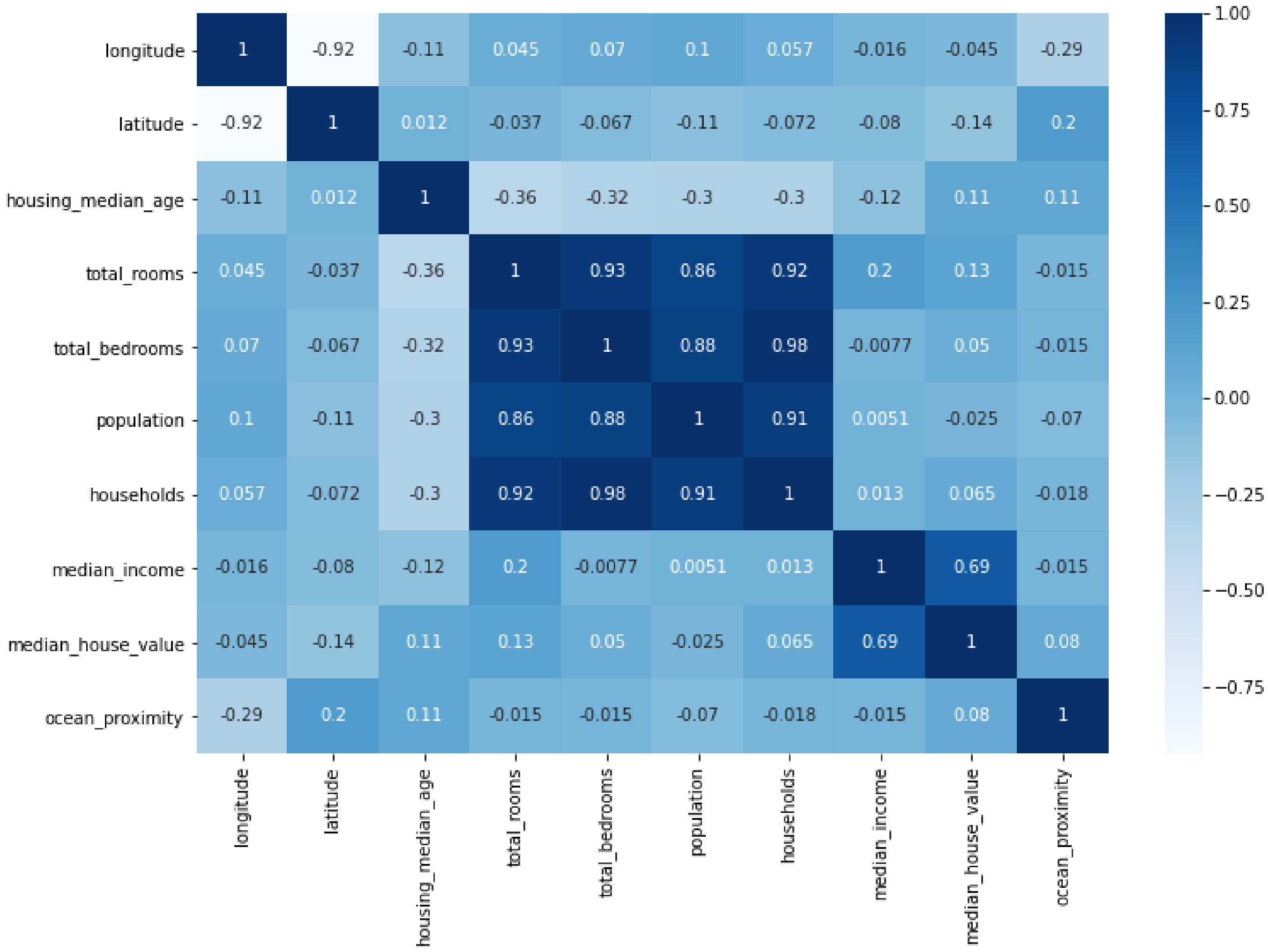
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(data[['housing_median_age',        'total_rooms',  'total_bedrooms',       'population',   'households',
'median_income' ]], diag_kind="kde")
plt.show()
```



In [ ]:
```
# Some features are significantly skewed
data.hist(bins=50, figsize=(10, 10))
plt.show()
```

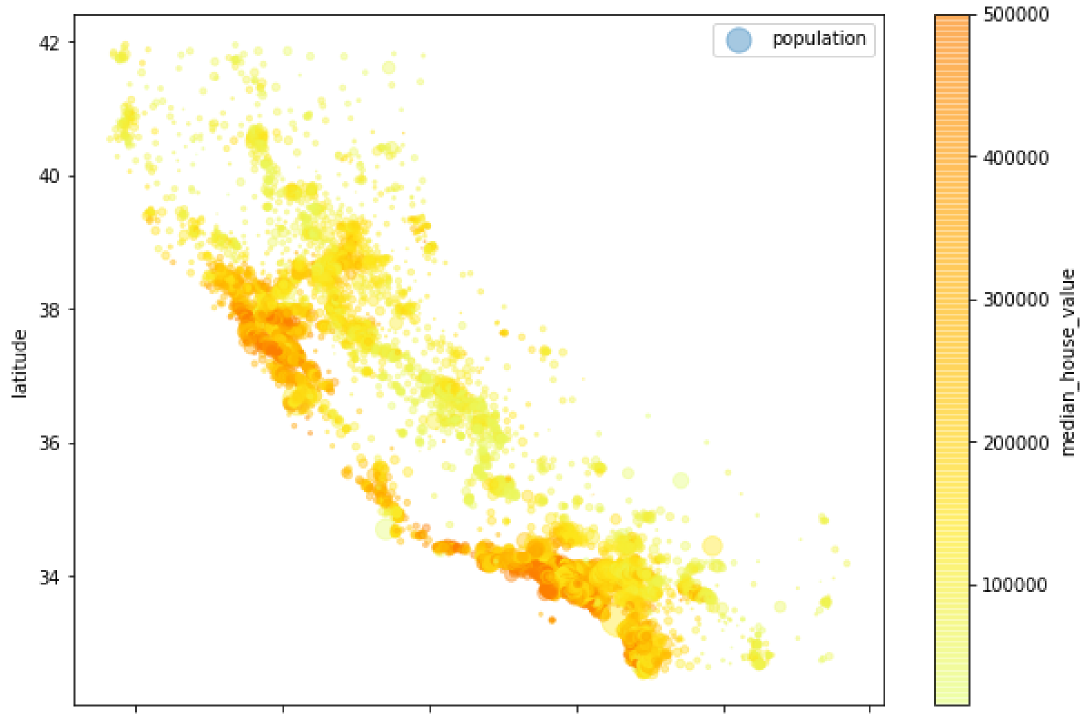


In [ ]:
```
import seaborn as sns
corr = pd.DataFrame(data).corr()
plt.rcParams['figure.figsize'] = (12, 8)
sns.heatmap(corr, xticklabels=corr.columns.values, yticklabels=corr.columns.values, annot=True,cmap='Blues')
plt.show()
```

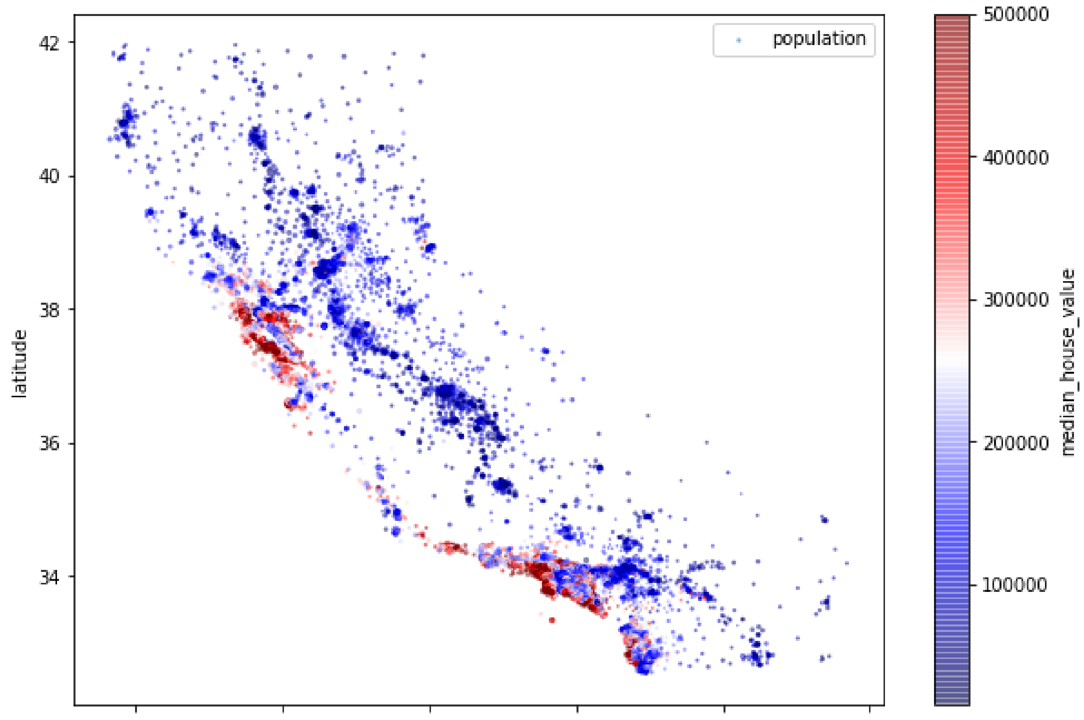


- House values and median income are correlated significantly.
- Population and households are highly correlated but not 100%.
- Highly correlated features: number of rooms, bedrooms, population and households.

In [ ]:
```
# s: Size represents population density by 100
# c: Median price is represented by color
data.plot(kind='scatter', x='longitude', y='latitude', alpha=0.4,
    s=data.population/100, label='population', figsize=(10,7),
    c='median_house_value', cmap=plt.get_cmap('Wistia'), colorbar=True)
plt.show()
```



In [ ]:
```
# s: Size represents population density by 100
# c: Median price is represented by color
data.plot(kind='scatter', x='longitude', y='latitude', alpha=0.4,
    s=data.housing_median_age/10, label='population', figsize=(10,7),
    c='median_house_value', cmap=plt.get_cmap('seismic'), colorbar=True)
plt.show()
```



Question-3: Split the dataset into training and testing set. Create a sequential model using RMSprop optimizer.

In [258]:
```
X_train, X_test, y_train,y_test=train_test_split(data.drop('median_house_value',axis=1),data.median_house_value, test_
size=0.3,random_state=101)
```

In [267]:
```
X_train.shape
```

Out[267]: (14303, 9)

In [279]:
```
# Building the model
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
def build_model():
  model = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=[len(X_train.keys())]),
    layers.Dense(128, activation='relu'),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model
```

In [280]:
```
model = build_model()
```

In [281]:
```
model.summary()
```

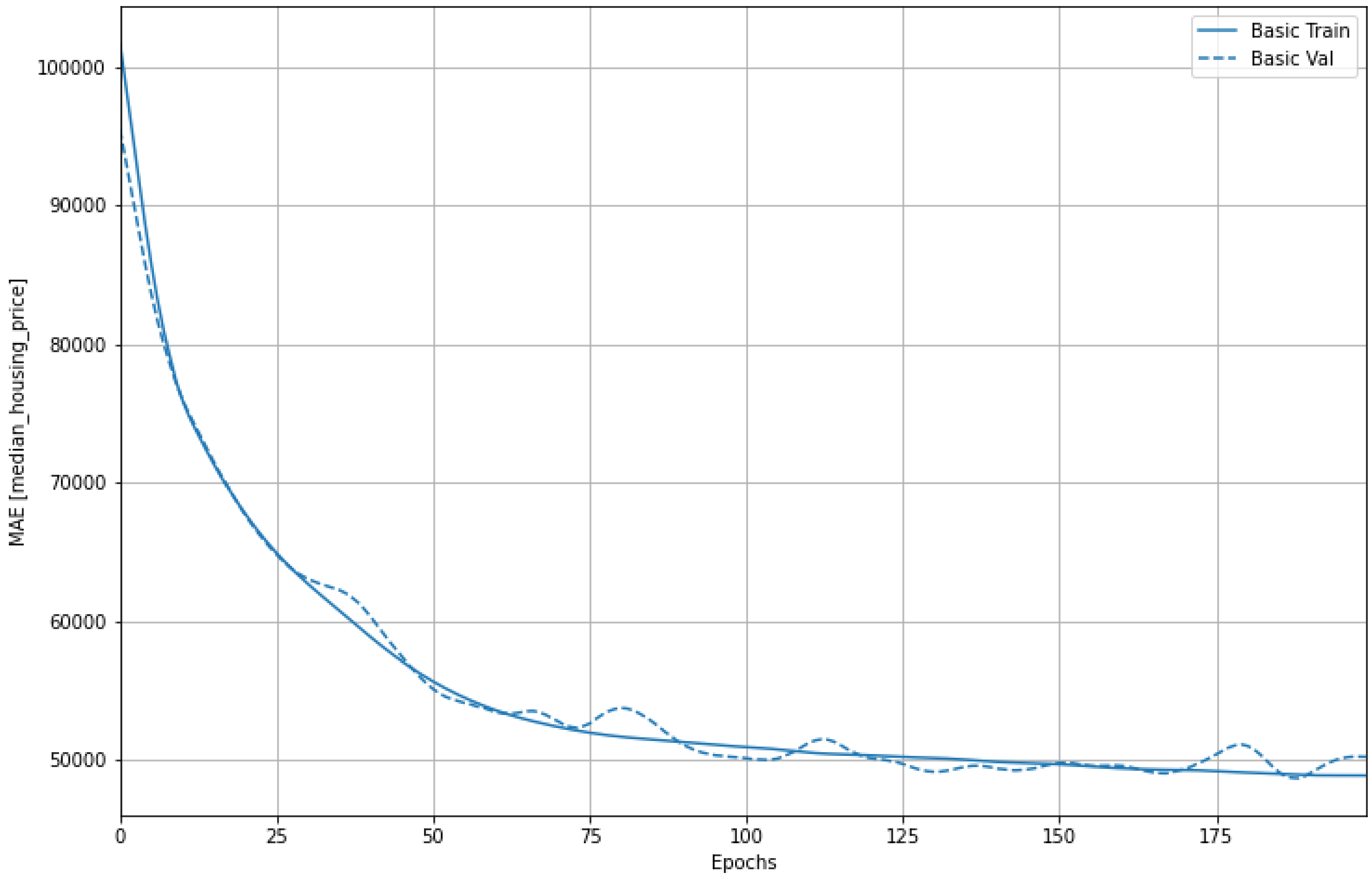
```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_38 (Dense)             (None, 128)               1280      
_________________________________________________________________
dense_39 (Dense)             (None, 128)               16512     
_________________________________________________________________
dense_40 (Dense)             (None, 1)                 129       
=================================================================
Total params: 17,921
Trainable params: 17,921
Non-trainable params: 0
_________________________________________________________________
```

Question-4: Fit the model for using EpochDots and plot the history of the model using HistoryPlotter.

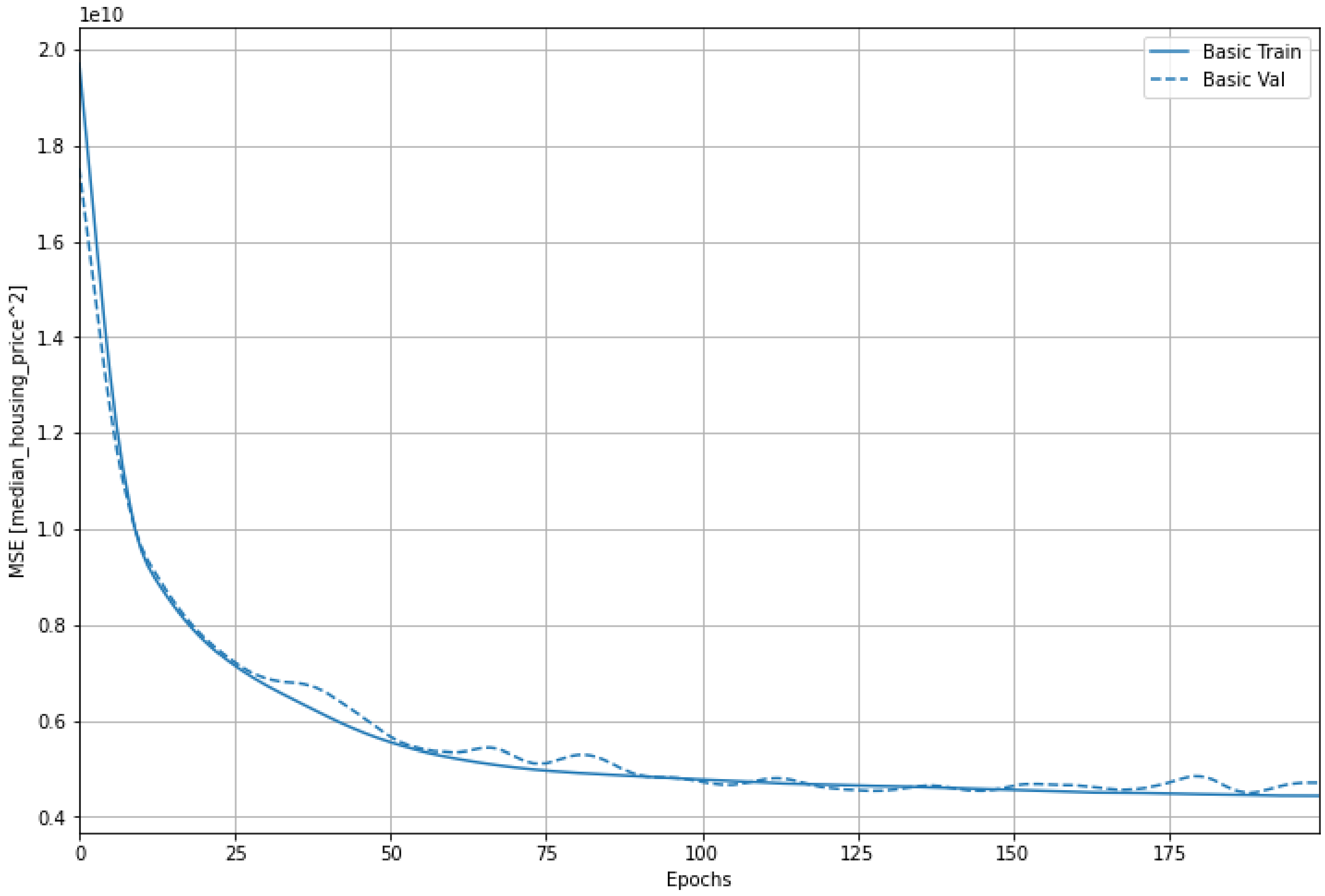
In [282]:
```
import tensorflow_docs as tfdocs
import tensorflow_docs.plots
import tensorflow_docs.modeling
EPOCHS = 200

history = model.fit(
  X_train, y_train,
epochs=EPOCHS, validation_split = 0.2, verbose=0,callbacks=[tensorflow_docs.modeling.EpochDots(20)])
```

```
Epoch: 0, loss:31375343616.0000,  mae:132564.3438,  mse:31375343616.0000,  val_loss:26157764608.0000,  val_mae:11610
1.9609,  val_mse:26157764608.0000,
....................
Epoch: 20, loss:7612354560.0000,  mae:67624.4453,  mse:7612354560.0000,  val_loss:7571301376.0000,  val_mae:66690.453
1,  val_mse:7571301376.0000,
....................
Epoch: 40, loss:6051584000.0000,  mae:58760.5117,  mse:6051584000.0000,  val_loss:6525977088.0000,  val_mae:59702.035
2,  val_mse:6525977088.0000,
....................
Epoch: 60, loss:5215568384.0000,  mae:53410.7031,  mse:5215568384.0000,  val_loss:5584291840.0000,  val_mae:54038.273
4,  val_mse:5584291840.0000,
....................
Epoch: 80, loss:4906635776.0000,  mae:51531.3164,  mse:4906635776.0000,  val_loss:5483291136.0000,  val_mae:56864.273
4,  val_mse:5483291136.0000,
....................
Epoch: 100, loss:4793686528.0000,  mae:50959.2148,  mse:4793686528.0000,  val_loss:4522398720.0000,  val_mae:48572.40
23,  val_mse:4522398720.0000,
....................
Epoch: 120, loss:4694982144.0000,  mae:50409.8789,  mse:4694982144.0000,  val_loss:4593113088.0000,  val_mae:50426.70
31,  val_mse:4593113088.0000,
....................
Epoch: 140, loss:4607792128.0000,  mae:49734.7070,  mse:4607792128.0000,  val_loss:4381277696.0000,  val_mae:47699.81
25,  val_mse:4381277696.0000,
....................
Epoch: 160, loss:4513398784.0000,  mae:49398.8086,  mse:4513398784.0000,  val_loss:4501460992.0000,  val_mae:50465.92
97,  val_mse:4501460992.0000,
....................
Epoch: 180, loss:4501150720.0000,  mae:49219.4023,  mse:4501150720.0000,  val_loss:5467736576.0000,  val_mae:58250.39
06,  val_mse:5467736576.0000,
....................
```

In [284]:
```
plotter = tfdocs.plots.HistoryPlotter(smoothing_std=2)
plotter.plot({'Basic': history}, metric = "mae")

plt.ylabel('MAE [median_housing_price]')
plt.show()
```

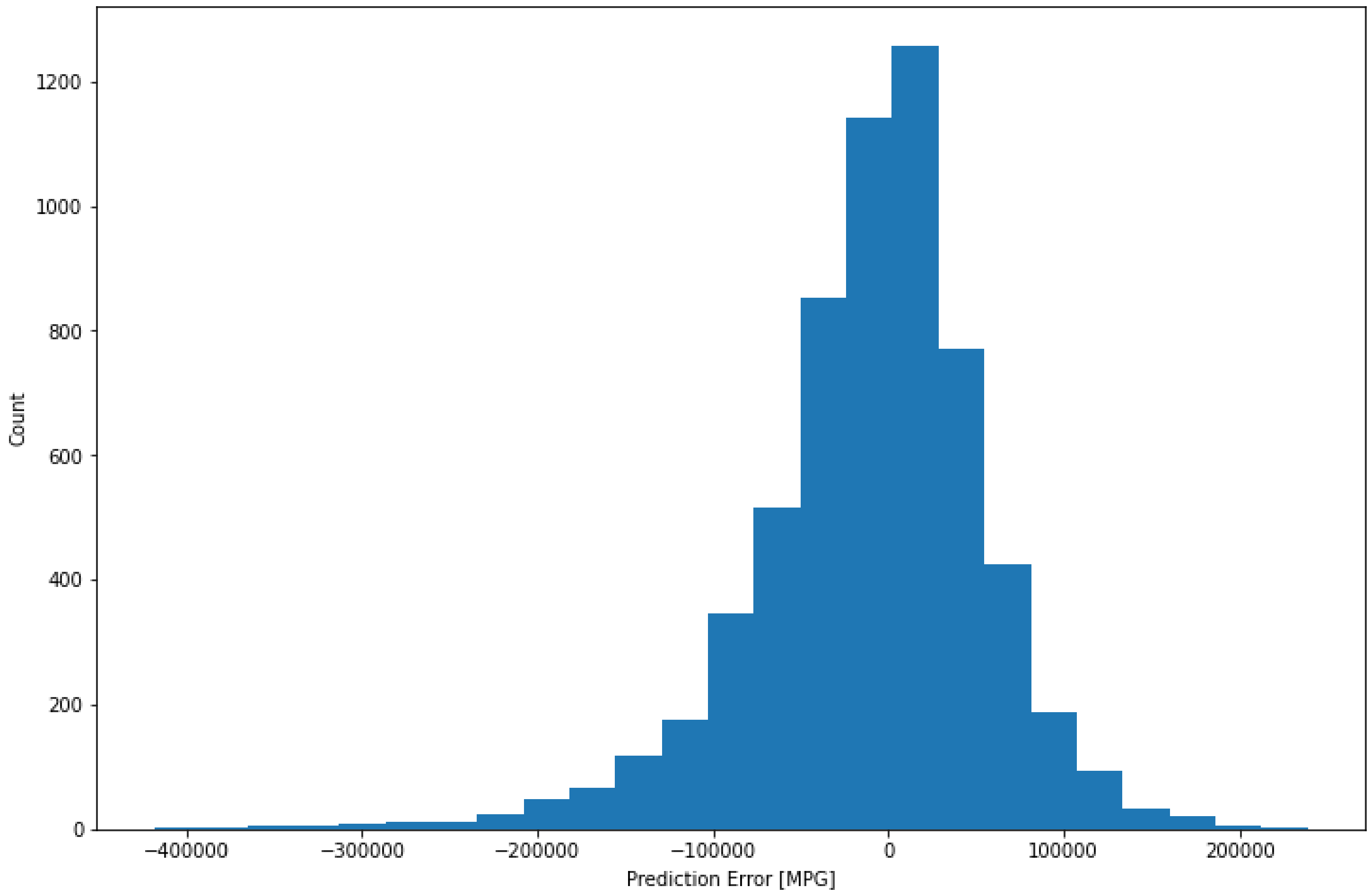


In [285]:
```
plotter.plot({'Basic': history}, metric = "mse")
plt.ylabel('MSE [median_housing_price^2]')
plt.show()
```



Question-5: Plot a histogram of errors.

In [294]:
```
test_predictions = model.predict(X_test).flatten()
error = test_predictions - y_test
plt.hist(error, bins = 25)
plt.xlabel("Prediction Error")
_ = plt.ylabel("Count")
```



The error is gaussian in nature, we can reduce it using more epochs or normalizing the data.

CNN

CNN is a Deep learning classification algorithm that takes an image as an input, extract features, and assign importance (weights and biases) to various aspects/ objects in the picture, to differentiate one from the other

Layers in CNN:

1. Convolutional Layer- Filtering is done to identify a particular feature for trying every possible position
2. ReLu- Remove every negative value from the filtered images and replace them with zero's
3. Pooling Layer- Reduce the size of the data
4. Fully Connected Layer (Dense)

OpenCV

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000.

- It is a Python library which is designed to solve computer vision problems. OpenCV was originally developed in 1999 by Intel but later it was supported by Willow Garage.
- It supports a wide variety of programming languages such as C++, Python, Java etc. Support for multiple platforms including Windows, Linux, and MacOS.
- OpenCV Python is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays. This makes it easier to integrate it with other libraries which use NumPy. For example, libraries such as SciPy and Matplotlib.

Traffic Sign Classification using CNN on Tensorflow 2.0

Caltech Automobiles is a famous car manufacturing industry. Although automobile popularity has brought considerable convenience to people, it has also caused numerous traffic safety issues that can not be ignored, such as congestion and frequent road accidents.

Traffic safety issues are caused mainly by driver-related subjective reasons, such as inattention, improper driving, and failing to comply with traffic rules.

Hence, to avoid these issues in the future CEO of Caltech decides to build smart cars (Self-driving cars)

Self-driving technology can assist or even complete the driving operation independently, which is of considerable importance for relieving the human body and significantly reducing the incidence of accidents.

Problem Statement:

Detection and recognition of traffic signs are crucial for the development of self-driving cars, which have a direct impact on driving behaviors.

Self-driving cars use a vehicle-mounted camera to obtain real and practical road traffic information; they can also recognize and understand traffic signs in real-time in road scenes to provide smart vehicles with correct command output and reasonable movement control, which can considerably improve the performance and safety of automatic driving.

So, The CEO of Caltech decides to hire an Analyst who can build a CNN model which Detects and classifies the Traffic signals according to its labels, for his new Self-driving Cars.

Tasks to be performed:

Our objective is to build a CNN model which classifies the Traffic signals and predicts them correctly, In order to do that we need to perform the below tasks:

- Load the data (pickle files) and segregate them into features and labels. Print their shapes- Beginner
- Visualize the segregatted images along with their labels- Beginner
- Convert the images to grayscale and print their shape- Intermediate
- Normalize the greyscaled images and visualize them- Intermediate
- Build a CNN model using Sequential API- Advance
- Print and check the model Summary- Beginner
- Compile the model using Adam optimizer, sparse crossentropy loss and calculate its accuracy metrics- Intermediate
- Fit and train the model with 15 epochs and 500 batchsize- Advance
- Calculate and print the accuracy score for test data- Intermediate
- Visualize Training and validation loss and write your inference- Advance
- Visualize Training and validation accuracy and write your inference- Advance
- Calculate the classification report for each class- Intermediate
- Visualize the predicted images and write your inference- Advance

Dataset Description:

The dataset consists of 43 different classes of images. Classes are as listed below:

- 0 = Speed limit (20km/h)
- 1 = Speed limit (30km/h)
- 2 = Speed limit (50km/h)
- 3 = Speed limit (60km/h)
- 4 = Speed limit (70km/h)
- 5 = Speed limit (80km/h)
- 6 = End of speed limit (80km/h)
- 7 = Speed limit (100km/h)
- 8 = Speed limit (120km/h)
- 9 = No passing
- 10 = No passing for vehicles over 3.5 metric tons
- 11 = Right-of-way at the next intersection
- 12 = Priority road
- 13 = Yield
- 14 = Stop
- 15 = No vehicles
- 16 = Vehicles over 3.5 metric tons prohibited
- 17 = No entry
- 18 = General caution
- 19 = Dangerous curve to the left
- 20 = Dangerous curve to the right
- 21 = Double curve
- 22 = Bumpy road
- 23 = Slippery road
- 24 = Road narrows on the right
- 25 = Road work
- 26 = Traffic signals
- 27 = Pedestrians
- 28 = Children crossing
- 29 = Bicycles crossing
- 30 = Beware of ice/snow
- 31 = Wild animals crossing
- 32 = End of all speed and passing limits
- 33 = Turn right ahead
- 34 = Turn left ahead
- 35 = Ahead only
- 36 = Go straight or right
- 37 = Go straight or left
- 38 = Keep right
- 39 = Keep left
- 40 = Roundabout mandatory
- 41 = End of no passing
- 42 = End of no passing by vehicles over 3.5 metric tons

Topics Covered

- Tensorflow 2.0
- CNN

Import Tensorflow and check for its version

In [1]:
```
# Importing tensorflow and checking for the version
import tensorflow as tf
print(tf.__version__)
```

```
2.2.0
```

We can see that we are using the latest version of tensorflow

In [2]:
```
# importing the required libraries
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import pandas as pd
import seaborn as sns
import pickle
import random
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprec
ated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [3]:
```
!wget https://www.dropbox.com/s/n2wzd6k7t9u6yyx/valid.p
```

```
--2020-07-17 05:14:00--  https://www.dropbox.com/s/n2wzd6k7t9u6yyx/valid.p
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1, 2620:100:6021:1::a27d:4101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/n2wzd6k7t9u6yyx/valid.p [following]
--2020-07-17 05:14:00--  https://www.dropbox.com/s/raw/n2wzd6k7t9u6yyx/valid.p
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc443a1cd852d899cf41eda364ec.dl.dropboxusercontent.com/cd/0/inline/A7pXHRcLqZYPzY9bGsutb8XmEbf9Hzmb
uZWuAnzC0t3cd7Pt5fK23gmcgg5iYLV5DvR0wYf2zLmSHQuwMv90Jy-YTneaTCVDwgzI9NHGPJekaDQb128D2_IllK7blEnnCr0/file# [following]
--2020-07-17 05:14:00--  https://uc443a1cd852d899cf41eda364ec.dl.dropboxusercontent.com/cd/0/inline/A7pXHRcLqZYPzY9bG
sutb8XmEbf9HzmbuZWuAnzC0t3cd7Pt5fK23gmcgg5iYLV5DvR0wYf2zLmSHQuwMv90Jy-YTneaTCVDwgzI9NHGPJekaDQb128D2_IllK7blEnnCr0/fi
le
Resolving uc443a1cd852d899cf41eda364ec.dl.dropboxusercontent.com (uc443a1cd852d899cf41eda364ec.dl.dropboxusercontent.
com)... 162.125.65.15, 2620:100:6021:15::a27d:410f
Connecting to uc443a1cd852d899cf41eda364ec.dl.dropboxusercontent.com (uc443a1cd852d899cf41eda364ec.dl.dropboxusercont
ent.com)|162.125.65.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13578712 (13M) [text/plain]
Saving to: ‘valid.p’

valid.p             100%[===================>]  12.95M  24.0MB/s    in 0.5s

2020-07-17 05:14:02 (24.0 MB/s) - ‘valid.p’ saved [13578712/13578712]
```

In [4]:
```
!wget https://www.dropbox.com/s/5qxezu9azevja57/train.p
```

```
--2020-07-17 05:14:06--  https://www.dropbox.com/s/5qxezu9azevja57/train.p
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1, 2620:100:6021:1::a27d:4101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/5qxezu9azevja57/train.p [following]
--2020-07-17 05:14:07--  https://www.dropbox.com/s/raw/5qxezu9azevja57/train.p
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucb87124081e3cd90c72b44eecb7.dl.dropboxusercontent.com/cd/0/inline/A7o6Dxn_zxfYDUpHc-YujuJ1nK5ahk1t
_mOpwYl3ULEmPKqVX2uoveDStqv4lImGKRBJkVRG18cASo-nXjoySKHsxLjNgUFKj8h5sTfrdFbTisVWBRQy1Z6gEPkzEofTLss/file# [following]
--2020-07-17 05:14:07--  https://ucb87124081e3cd90c72b44eecb7.dl.dropboxusercontent.com/cd/0/inline/A7o6Dxn_zxfYDUpHc
-YujuJ1nK5ahk1t_mOpwYl3ULEmPKqVX2uoveDStqv4lImGKRBJkVRG18cASo-nXjoySKHsxLjNgUFKj8h5sTfrdFbTisVWBRQy1Z6gEPkzEofTLss/fi
le
Resolving ucb87124081e3cd90c72b44eecb7.dl.dropboxusercontent.com (ucb87124081e3cd90c72b44eecb7.dl.dropboxusercontent.
com)... 162.125.65.15, 2620:100:6021:15::a27d:410f
Connecting to ucb87124081e3cd90c72b44eecb7.dl.dropboxusercontent.com (ucb87124081e3cd90c72b44eecb7.dl.dropboxusercont
ent.com)|162.125.65.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 107146452 (102M) [text/plain]
Saving to: ‘train.p’

train.p             100%[===================>] 102.18M  25.2MB/s    in 4.1s

2020-07-17 05:14:12 (25.2 MB/s) - ‘train.p’ saved [107146452/107146452]
```

In [5]:
```
!wget https://www.dropbox.com/s/zi7honh03yr85ns/test.p
```

```
--2020-07-17 05:14:18--  https://www.dropbox.com/s/zi7honh03yr85ns/test.p
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1, 2620:100:6021:1::a27d:4101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/zi7honh03yr85ns/test.p [following]
--2020-07-17 05:14:19--  https://www.dropbox.com/s/raw/zi7honh03yr85ns/test.p
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc6e9b350f003f301122e2898f2c.dl.dropboxusercontent.com/cd/0/inline/A7r6KfphukhEUD0NTRd4WstqTn6fIuZa
qcmdf5IiopRBXIU6K89zrWU01HSLVjP_g1LrnbCchjrhin2m1EHarjeDBdGtpuZKq-dafoVL7ombdrO0YNIuIRvbWW-8NNqVHYE/file# [following]
--2020-07-17 05:14:19--  https://uc6e9b350f003f301122e2898f2c.dl.dropboxusercontent.com/cd/0/inline/A7r6KfphukhEUD0NT
Rd4WstqTn6fIuZaqcmdf5IiopRBXIU6K89zrWU01HSLVjP_g1LrnbCchjrhin2m1EHarjeDBdGtpuZKq-dafoVL7ombdrO0YNIuIRvbWW-8NNqVHYE/fi
le
Resolving uc6e9b350f003f301122e2898f2c.dl.dropboxusercontent.com (uc6e9b350f003f301122e2898f2c.dl.dropboxusercontent.
com)... 162.125.65.15, 2620:100:6021:15::a27d:410f
Connecting to uc6e9b350f003f301122e2898f2c.dl.dropboxusercontent.com (uc6e9b350f003f301122e2898f2c.dl.dropboxusercont
ent.com)|162.125.65.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38888118 (37M) [text/plain]
Saving to: ‘test.p’

test.p              100%[===================>]  37.09M  25.7MB/s    in 1.4s

2020-07-17 05:14:21 (25.7 MB/s) - ‘test.p’ saved [38888118/38888118]
```

Question-1:

Load the data (pickle files) and segregate them into features and labels. Print their shapes

pickle file or (.p) file is also known as pickle file created by pickle (python object serialization library) module, which is used to convert Python objects to a Byte representation for disk storage or network transfer

The use of pickling and unpickling is widespread in real world sceanario as it allows us to easily transfer data from one server / system to another, and then store it in a file or database.

In [6]:
```
# load the data and store them in train, test and valid variables respectively
train = pickle.load(open('/content/train.p','rb'))
test = pickle.load(open('/content/test.p','rb'))
valid = pickle.load(open('/content/valid.p','rb'))
```

In [7]:
```
# segregate the data into features and labels
x_train, y_train= train['features'], train['labels']
x_validation, y_validation= valid['features'], valid['labels']
x_test, y_test= test['features'], test['labels']
```

In [8]:
```
# print the shape of data
print(x_train.shape)
print(x_validation.shape)
print(x_test.shape)
```

```
(34799, 32, 32, 3)
(4410, 32, 32, 3)
(12630, 32, 32, 3)
```

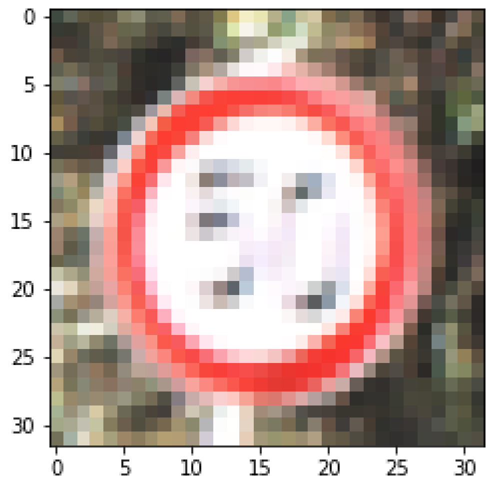
In [9]:
```
# print the shape of data
print(y_train.shape)
print(y_validation.shape)
print(y_test.shape)
```

```
(34799,)
(4410,)
(12630,)
```

Question-2:

Visualize the segregated images along with their labels

```
In [10]: i = np.random.randint(1, len(x_train))
         plt.imshow(x_train[i])
         y_train[i]

Out[10]: 2
```



Here we can observe that label number is 2 which means the sign indicates **Speed limit (50km/h)**

Note: As we are using random images output will change everytime we run the code. This particular label is with respect to above output obtained

In [11]:
```
# Let's view more images in a grid format
# Define the dimensions of the plot grid
W_grid = 5
L_grid = 5

# subplot return the figure object and axes object
# we can use the axes object to plot specific figures at various locations

fig, axes = plt.subplots(L_grid, W_grid, figsize = (10,10))

axes = axes.ravel() # flaten the 5 x 5 matrix into 25 array

n_training = len(x_train) # get the length of the training dataset

# Select a random number from 0 to n_training
# create evenly spaces variables
for i in np.arange(0, W_grid * L_grid):
    # Select a random number
    index=np.random.randint(0, n_training)
    # read and display an image with the selected index
    axes[i].imshow(x_train[index])
    axes[i].set_title(y_train[index],fontsize= 15)
    axes[i].axis('off')

plt.subplots_adjust(hspace=0.4 )
```

Each label represents each class which is described in the dataset

Question-3:

Convert the images to grayscale and print their shape

In [12]:
```
# we shuffle the data to consider the data randomly
from sklearn.utils import shuffle
x_train, y_train = shuffle(x_train, y_train)
```

In [13]:
```
# Converting the colored images to grey scale images
x_train_grey= np.sum(x_train/3, axis=3, keepdims=True)
x_test_grey= np.sum(x_test/3, axis=3, keepdims=True)
x_valid_grey= np.sum(x_validation/3, axis=3, keepdims=True)

# printing their shapes
print(x_train_grey.shape)
print(x_test_grey.shape)
print(x_valid_grey.shape)
```

```
(34799, 32, 32, 1)
(12630, 32, 32, 1)
(4410, 32, 32, 1)
```

We can observe that input shape dimension was 3 (RGB) but now we have only 1 which represents it is converted to grey scale

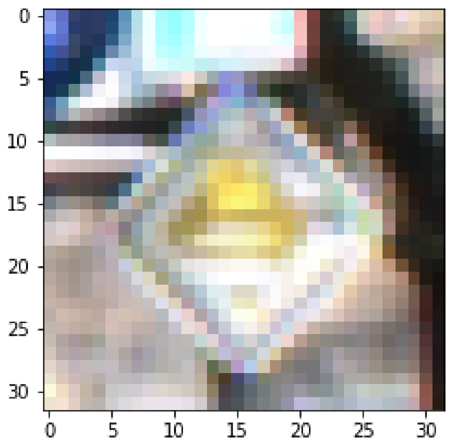
Question-4:

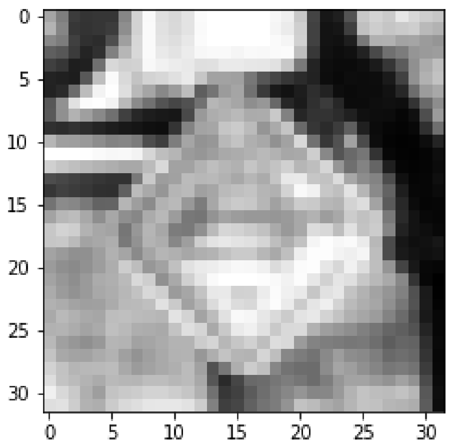
Normalize the greyscaled images and visualize them

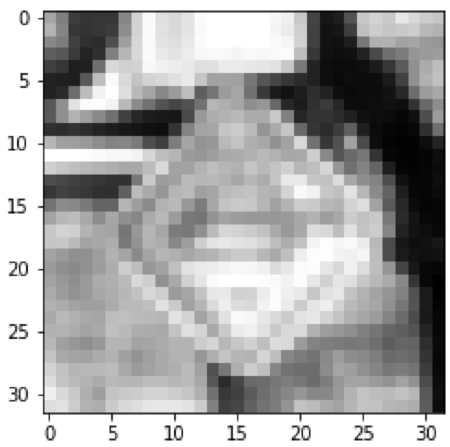
In [14]:
```
# Normalizing the data
x_train_grey_norm= (x_train_grey-255)/255
x_test_grey_norm= (x_test_grey-255)/255
x_valid_grey_norm= (x_valid_grey-255)/255
```

In [15]:
```
# visualizing the normalized data
i = random.randint(1, len(x_train_grey))
plt.imshow(x_train[i])
plt.figure()
plt.imshow(x_train_grey[i].squeeze(), cmap = 'gray')
plt.figure()
plt.imshow(x_train_grey_norm[i].squeeze(), cmap = 'gray')
```

Out[15]: <matplotlib.image.AxesImage at 0x7f1e7ee2b550>







We can observe that normal image is converted into grey scale and then normalized.

Now, this normalized data is our input to the model

Question-5:

Build a CNN model using Sequential API

- Define convolutional neural network in the model
- add() - Helps to add layers in the model
- Conv2D() - Convolutional layer (to extract features from the images)
- Conv2D(6,(5,5),input_shape=(32,32,1))
  - 6 - Take 6 features from the given image
  - (5,5) - Metrics size of the images(5*5)
  - input_shape = image size (32,32,1)
- Activation function (relu) is added to remove the negative values
- Dropout(0.2) used to deactivate 20% neurons randomly to prevent overfitting

Note:

This is one of the solution with the below mentioned Neurons and activation functions. You can always try out with different number of Neurons and activation function which might yeild you even better results

In [16]:
```
from tensorflow.keras import datasets, layers, models

model=models.Sequential()
model.add(layers.Conv2D(6, (5,5), activation='relu', input_shape=(32,32,1))) #6 neurons with 5*5 filter, relu is used
 to remove the negative values
model.add(layers.MaxPooling2D()) #MaxPooling2D helps to reduce the size of the data

model.add(layers.Dropout(0.2)) # to deactivate 20% neurons randomly to prevent overfitting

model.add(layers.Conv2D(16, (5,5), activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Flatten())#Converts multi dimensional array to 1D channel
model.add(layers.Dense(120, activation='relu'))# input layer with 120 neurons
model.add(layers.Dense(84, activation='relu'))# hidden layer with 84 neurons
model.add(layers.Dense(43, activation='softmax'))# output layer with 43 neurons
```

Question-6

Print and check the model Summary

In [18]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 6)         156
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 6)         0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 120)               48120
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_2 (Dense)              (None, 43)                3655
=================================================================
Total params: 64,511
Trainable params: 64,511
Non-trainable params: 0
_________________________________________________________________
```

Above summary tells us,

- After pooling and dropout functions our input neurons are 400
- We have 120 and 84 hidden neurons
- Our output has 43 neurons because we have 43 labels
- Total trainable parameters are 64,511

Question-7

Compile the model using Adam optimizer, sparse crossentropy loss and calculate its accuracy metrics

- Adam is an optimisation algorithm that can be used to adjust network weights iteratively based on training data instead of the traditional stochastic gradient descent method.
- Sparse categorical crossentropy is used because here we have multi-class classification problem, the labels are mutually exclusive for each data, meaning each data entry can only belong to one class
- Accuracy is the metrics we are using to evaluate the model

In [19]:
```
model.compile(optimizer='Adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Question-8:

Fit and train the model with 15 epochs and 500 batchsize

- Batch size refers to the number of training examples utilized in one iteration
- Epoch is the training samples pass through the learning algorithm simultaneously before weights are updated

In [20]:
```
history= model.fit(x_train_grey_norm,
                   y_train,
                   batch_size=500,
                   epochs=15,
                   verbose=1,
                   validation_data=(x_valid_grey_norm,y_validation))
```

```
Epoch 1/15
70/70 [==============================] - 1s 11ms/step - loss: 3.4131 - accuracy: 0.1017 - val_loss: 3.2052 - val_accu
racy: 0.1760
Epoch 2/15
70/70 [==============================] - 1s 8ms/step - loss: 2.4397 - accuracy: 0.3500 - val_loss: 1.9912 - val_accur
acy: 0.4662
Epoch 3/15
70/70 [==============================] - 1s 7ms/step - loss: 1.5341 - accuracy: 0.5706 - val_loss: 1.3985 - val_accur
acy: 0.6245
Epoch 4/15
70/70 [==============================] - 1s 8ms/step - loss: 1.1377 - accuracy: 0.6767 - val_loss: 1.1160 - val_accur
acy: 0.7079
Epoch 5/15
70/70 [==============================] - 1s 8ms/step - loss: 0.9102 - accuracy: 0.7440 - val_loss: 0.9976 - val_accur
acy: 0.7454
Epoch 6/15
70/70 [==============================] - 1s 8ms/step - loss: 0.7613 - accuracy: 0.7873 - val_loss: 0.9116 - val_accur
acy: 0.7571
Epoch 7/15
70/70 [==============================] - 1s 8ms/step - loss: 0.6408 - accuracy: 0.8216 - val_loss: 0.8237 - val_accur
acy: 0.7993
Epoch 8/15
70/70 [==============================] - 1s 8ms/step - loss: 0.5497 - accuracy: 0.8502 - val_loss: 0.7474 - val_accur
acy: 0.7980
Epoch 9/15
70/70 [==============================] - 1s 8ms/step - loss: 0.4746 - accuracy: 0.8688 - val_loss: 0.7047 - val_accur
acy: 0.8286
Epoch 10/15
70/70 [==============================] - 1s 8ms/step - loss: 0.4184 - accuracy: 0.8848 - val_loss: 0.6343 - val_accur
acy: 0.8397
Epoch 11/15
70/70 [==============================] - 1s 8ms/step - loss: 0.3661 - accuracy: 0.9012 - val_loss: 0.6137 - val_accur
acy: 0.8562
Epoch 12/15
70/70 [==============================] - 1s 8ms/step - loss: 0.3372 - accuracy: 0.9066 - val_loss: 0.5894 - val_accur
acy: 0.8558
Epoch 13/15
70/70 [==============================] - 1s 8ms/step - loss: 0.3017 - accuracy: 0.9169 - val_loss: 0.6265 - val_accur
acy: 0.8567
Epoch 14/15
70/70 [==============================] - 1s 8ms/step - loss: 0.2772 - accuracy: 0.9245 - val_loss: 0.6043 - val_accur
acy: 0.8621
Epoch 15/15
70/70 [==============================] - 1s 8ms/step - loss: 0.2539 - accuracy: 0.9306 - val_loss: 0.5964 - val_accur
acy: 0.8687
```

Here we can observe that our training data accuracy is 89% and validation accuracy is 86%

Question-9

Calculate and print the accuracy score for test data

In [21]:
```
score = model.evaluate(x_test_grey_norm, y_test)
print('Test Accuracy: {}'.format(score[1]))
```

```
395/395 [==============================] - 1s 2ms/step - loss: 0.7566 - accuracy: 0.8755
Test Accuracy: 0.8755344152450562
```

We can see that Test accuracy is almost 85%

Question-10

Visualize Training and validation loss and write your inference

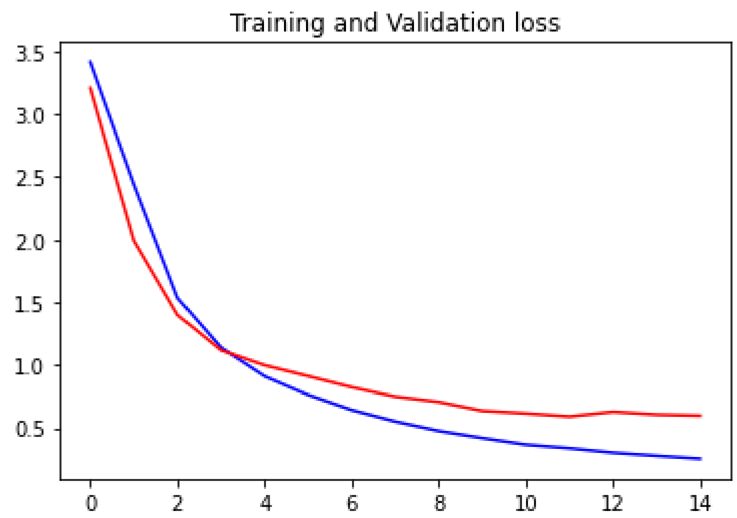
In [22]:
```
history.history.keys()
```

Out[22]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [23]:
```
# store the history values into accuracy, val_accuracy, loss and val_loss
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [24]:
```
epochs= range(len(accuracy))
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
```

Out[24]:
```
Text(0.5, 1.0, 'Training and Validation loss')
```



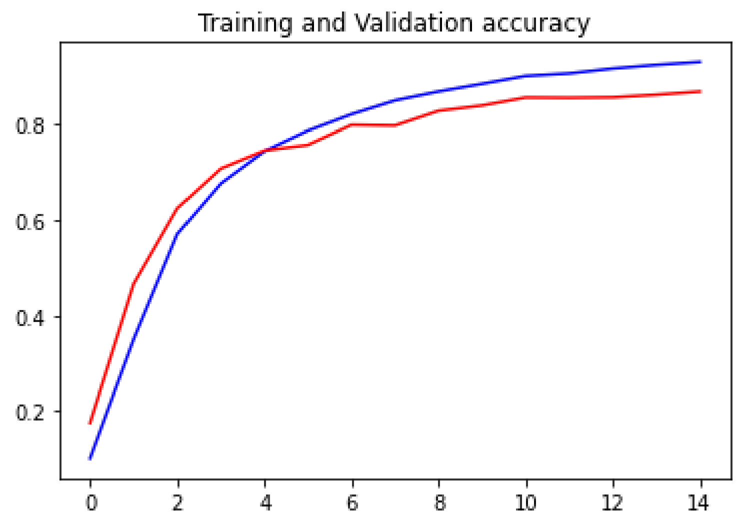
We can observe that both the losses are almost same, after some time validation loss reached saturation

Question-11

Visualize Training and validation accuracy and write your inference

In [25]:
```
epochs= range(len(accuracy))
plt.plot(epochs, accuracy, 'b', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'r', label='Validation accuracy')
plt.title('Training and Validation accuracy')
```

Out[25]:
```
Text(0.5, 1.0, 'Training and Validation accuracy')
```



We can observe that both the accuracies are almost same, after some time validation accuracy reached saturation

Question-12

Calculate the classification report for each class

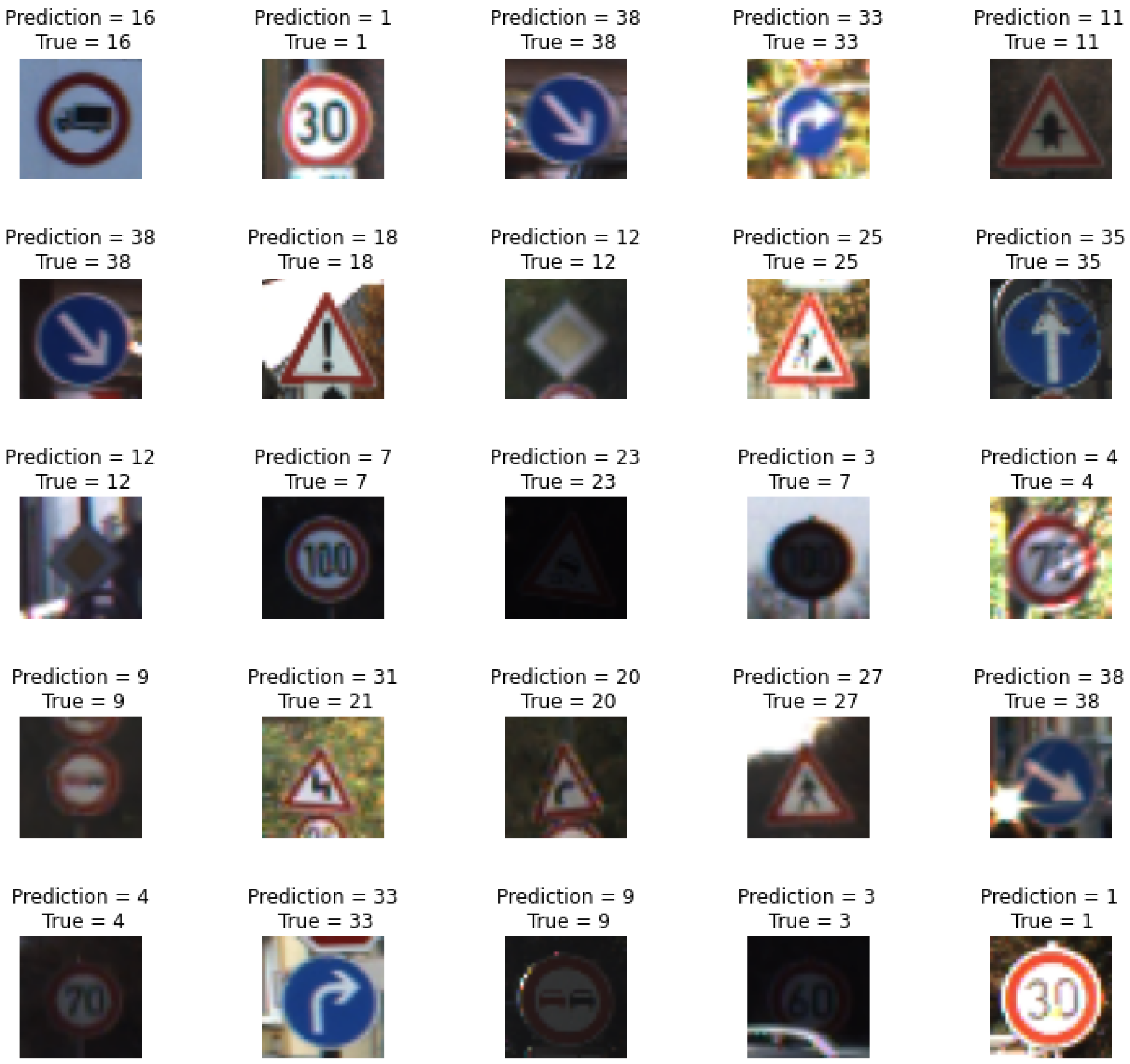
In [27]:
```
from sklearn.metrics import accuracy_score, classification_report
predicted_classes = model.predict_classes(x_test_grey_norm)
y_true = y_test
print(classification_report(y_test,predicted_classes))
```

```
WARNING:tensorflow:From <ipython-input-27-b3c2125471ce>:2: Sequential.predict_classes (from tensorflow.python.keras.e
ngine.sequential) is deprecated and will be removed after 2021-01-01.
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g.
if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does bina
ry classification   (e.g. if it uses a `sigmoid` last-layer activation).
              precision    recall  f1-score   support

           0       0.78      0.48      0.60        60
           1       0.82      0.94      0.88       720
           2       0.92      0.92      0.92       750
           3       0.83      0.92      0.87       450
           4       0.92      0.87      0.90       660
           5       0.83      0.83      0.83       630
           6       0.93      0.81      0.87       150
           7       0.87      0.76      0.81       450
           8       0.77      0.90      0.83       450
           9       0.91      0.96      0.94       480
          10       0.95      0.95      0.95       660
          11       0.81      0.84      0.82       420
          12       0.96      0.96      0.96       690
          13       0.95      0.99      0.97       720
          14       0.93      0.89      0.91       270
          15       0.84      0.95      0.89       210
          16       1.00      0.97      0.98       150
          17       0.99      0.91      0.95       360
          18       0.86      0.71      0.78       390
          19       0.84      0.63      0.72        60
          20       0.74      0.66      0.69        90
          21       0.92      0.50      0.65        90
          22       0.97      0.95      0.96       120
          23       0.76      0.63      0.69       150
          24       0.55      0.31      0.40        90
          25       0.91      0.89      0.90       480
          26       0.68      0.92      0.78       180
          27       0.62      0.47      0.53        60
          28       0.75      0.87      0.80       150
          29       0.61      0.79      0.69        90
          30       0.56      0.45      0.50       150
          31       0.79      0.89      0.84       270
          32       0.72      0.95      0.82        60
          33       0.93      0.91      0.92       210
          34       0.94      0.97      0.96       120
          35       0.96      0.86      0.91       390
          36       0.94      0.85      0.89       120
          37       0.84      0.98      0.91        60
          38       0.92      0.93      0.92       690
          39       0.95      0.89      0.92        90
          40       0.63      0.53      0.58        90
          41       0.96      0.80      0.87        60
          42       0.94      0.93      0.94        90

    accuracy                           0.88     12630
   macro avg       0.85      0.82      0.82     12630
weighted avg       0.88      0.88      0.87     12630
```

Question-13

Visualize the predicted images and write your inference

In [ ]:
```
L = 5
W = 5

fig, axes = plt.subplots(L, W, figsize = (12, 12))
axes = axes.ravel()

for i in np.arange(0, L*W):
    axes[i].imshow(x_test[i])
    axes[i].set_title('Prediction = {}\n True = {}'.format(predicted_classes[i], y_true[i]))
    axes[i].axis('off')

plt.subplots_adjust(wspace = 1)
```



Here we can observe that out of 25 images, label with numbers 7 and 21 are predicted wrongly as 3 and 21 respectively

Real time edge Detection using OpenCV

Fargo (A MNC company) decides to hire an Analyst (Fresher), But Fargo expects the Analyst to have knowledge on Open CV. Hence they decided to include a Open CV question in the interview process

Problem Statement

Fargo wants an Analyst who can build a model which can detect the edges in real time

They are not worried about all the intricate details of an image, but rather only care about the overall shape in real time

Tasks to be performed

Our objective is to build a model using OpenCV which edge detection in real-time correctly, In order to do that we need to perform the below tasks:

- Capture the video using Videocapture function- Beginner
- Read the video captured above, convert it to HSV to and use canny detection algorithm to detect the edges in real-time. Visulize the same- Intermediate
- Release the resources after recording and destroy all windows- Beginner

Topics Covered:

OpenCV

Note: Run this code in Jupyter or in your local system, because colab will not support

In [ ]:
```
# import libraries of python OpenCV
import cv2
# np is an alias pointing to numpy library
import numpy as np
```

Question-1

Capture the video using Videocapture function

In [ ]:
```
# capture frames from a camera
cap = cv2.VideoCapture(0)
```

Question-2

Read the video captured above, convert it to HSV to and use canny detection algorithm to detect the edges in real-time. Visulize the same

Inside the while loop,

- We declare 2 variables **success, frame** where frame captures the sequesnce of images and success returns boolean values whether the image is captured or not
- In the next step we convert BGR to HSV using cvtcolor function
- We define the range for the color
- Display the image
- Use canny image detector to detect the image
- If you have to exit press **esc** key

In [ ]:
```
# loop runs if capturing has been initialized
while(1):

    # reads frames from a camera
    success, frames = cap.read()

    # converting BGR to HSV
    hsv = cv2.cvtColor(frames, cv2.COLOR_BGR2HSV)

    # define range of red color in HSV
    low_red = np.array([30,150,50])
    up_red = np.array([255,255,180])

    # create a red HSV colour boundary and
    # threshold HSV image
    boundary = cv2.inRange(hsv, low_red, up_red)

    # Bitwise-AND mask and original image
    res = cv2.bitwise_and(frames,frames, mask= boundary)

    # Display an original image
    cv2.imshow('Original',frames)

    # finds edges in the input image image and
    # marks them in the output map edges
    edge = cv2.Canny(frames,100,200)

    # Display edges in a frame
    cv2.imshow('Edges',edge)

    # Wait for Esc key to stop
    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break
```

Question-3

Release the resources after recording and destroy all windows

In [ ]:
```
# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```

Full Human Body Detection

Caltech Automobiles is a famous car manufacturing industry. Although automobile popularity has brought considerable convenience to people, it has also caused numerous traffic safety issues that can not be ignored, such as congestion and frequent road accidents.

Traffic safety issues are caused mainly by driver-related subjective reasons, such as inattention, improper driving, and failing to comply with traffic rules.

Hence, to avoid these issues in the future CEO of Caltech decides to build smart cars (Self-driving cars)

Self-driving technology can assist or even complete the driving operation independently, which is of considerable importance for relieving the human body and significantly reducing the incidence of accidents.

Problem Statement:

Human Detection are crucial for the development of self-driving cars, which have a direct impact on driving behaviors.

Self-driving cars use a vehicle-mounted camera to obtain real and practical human movement information; they can also recognize and understand humans in real-time in road scenes to provide smart vehicles with correct command output and reasonable movement control, which can considerably improve the performance and safety of automatic driving.

So, The CEO of Caltech decides to hire an Analyst who can build a model which Detects humans, for his new Self-driving Cars

But before implementing it directly, he wants to check on video captured previously

Tasks to be performed:

Our objective is to build a model using OpenCV which detects the pedestrians on road correctly, In order to do that we need to perform the below tasks:

- Capture the pre-stored video and haarcascade file- Beginner
- Read the video captured above, convert it to grey scale to detect the human using detectMultiScale. Visulize the same- Intermediate
- Release the resources after recording and destroy all windows- Beginner

Topics Covered:

OpenCV

Note: Run this code in Jupyter or in your local system, because colab will not support

In [ ]:
```
!wget https://www.dropbox.com/s/7msg6kqvspgsgkp/haarcascade_fullbody.xml?dl=0
```

```
--2020-06-24 06:45:03--  https://www.dropbox.com/s/7msg6kqvspgsgkp/haarcascade_fullbody.xml?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1, 2620:100:6021:1::a27d:4101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/7msg6kqvspgsgkp/haarcascade_fullbody.xml [following]
--2020-06-24 06:45:03--  https://www.dropbox.com/s/raw/7msg6kqvspgsgkp/haarcascade_fullbody.xml
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc15d951c91fa3b56ebd56dcab3e.dl.dropboxusercontent.com/cd/0/inline/A6Mj8yGVcFw7B3yK8PV2BopfCglXPBtg
QZugrTGi7ysr14SdmpLSC4Q4DO1sxtMEsCFCabh47ifDgjFVcLn2oSDzGApKEmR3QffV_VOnxzkGZoFrUjsA15MlzlbntwdXwe0/file# [following]
--2020-06-24 06:45:04--  https://uc15d951c91fa3b56ebd56dcab3e.dl.dropboxusercontent.com/cd/0/inline/A6Mj8yGVcFw7B3yK8
PV2BopfCglXPBtgQZugrTGi7ysr14SdmpLSC4Q4DO1sxtMEsCFCabh47ifDgjFVcLn2oSDzGApKEmR3QffV_VOnxzkGZoFrUjsA15MlzlbntwdXwe0/fi
le
Resolving uc15d951c91fa3b56ebd56dcab3e.dl.dropboxusercontent.com (uc15d951c91fa3b56ebd56dcab3e.dl.dropboxusercontent.
com)... 162.125.65.15, 2620:100:6021:15::a27d:410f
Connecting to uc15d951c91fa3b56ebd56dcab3e.dl.dropboxusercontent.com (uc15d951c91fa3b56ebd56dcab3e.dl.dropboxusercont
ent.com)|162.125.65.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 476827 (466K) [text/plain]
Saving to: ‘haarcascade_fullbody.xml?dl=0’

haarcascade_fullbod 100%[===================>] 465.65K  --.-KB/s    in 0.02s

2020-06-24 06:45:04 (25.4 MB/s) - ‘haarcascade_fullbody.xml?dl=0’ saved [476827/476827]
```

In [ ]:
```
!wget https://www.dropbox.com/s/slnlq6ouh9yieev/vtest.avi?dl=0
```

```
--2020-06-24 06:45:53--  https://www.dropbox.com/s/slnlq6ouh9yieev/vtest.avi?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1, 2620:100:6021:1::a27d:4101
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/slnlq6ouh9yieev/vtest.avi [following]
--2020-06-24 06:45:54--  https://www.dropbox.com/s/raw/slnlq6ouh9yieev/vtest.avi
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc5fb759c8ea2d994b62cc56ede1.dl.dropboxusercontent.com/cd/0/inline/A6NAZXVuwAXDpCr0NWySeBYNeQk5ept3
7eAJrjMqpHW1YsLeg0Dvr3aYlu9i68VqIy1TFWXx3tvcRj-Z-wphlZVTF9yepQPwXAcmNXncyh4SZ_tEhRWnUWPEY9LIABb1fj4/file# [following]
--2020-06-24 06:45:54--  https://uc5fb759c8ea2d994b62cc56ede1.dl.dropboxusercontent.com/cd/0/inline/A6NAZXVuwAXDpCr0N
WySeBYNeQk5ept37eAJrjMqpHW1YsLeg0Dvr3aYlu9i68VqIy1TFWXx3tvcRj-Z-wphlZVTF9yepQPwXAcmNXncyh4SZ_tEhRWnUWPEY9LIABb1fj4/fi
le
Resolving uc5fb759c8ea2d994b62cc56ede1.dl.dropboxusercontent.com (uc5fb759c8ea2d994b62cc56ede1.dl.dropboxusercontent.
com)... 162.125.65.15, 2620:100:6021:15::a27d:410f
Connecting to uc5fb759c8ea2d994b62cc56ede1.dl.dropboxusercontent.com (uc5fb759c8ea2d994b62cc56ede1.dl.dropboxusercont
ent.com)|162.125.65.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8131690 (7.8M) [video/x-msvideo]
Saving to: 'vtest.avi?dl=0'

vtest.avi?dl=0      100%[===================>]   7.75M  28.0MB/s    in 0.3s

2020-06-24 06:45:55 (28.0 MB/s) - 'vtest.avi?dl=0' saved [8131690/8131690]
```

In [ ]:
```
# import libraries of python OpenCV
import cv2
```

Question-1

Capture the pre-stored video and haarcascade file

Haar cascade files is a machine learning object detection algorithm used to identify objects in an image or video

You can download more haarcascade files from the official OpenCV github link given below,

OpenCV Github Link https://github.com/opencv (https://github.com/opencv)

In [ ]:
```
# captured video
cap = cv2.VideoCapture('/content/vtest.avi?dl=0')

#get the harcascade file
faceCascade = cv2.CascadeClassifier("/content/haarcascade_fullbody.xml?dl=0")
```

Question-2

Read the video captured above, convert it to grey scale to detect the pedestrian using detectMultiScale. Visulize the same

In [ ]:
```
while True:
    success, frame = cap.read() #frame variable will capture the Video & success variable will tell us whether it was
 captured successfully or not


    imgGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(imgGray, 1.1, 4) # imggray is the scalefactor, 1.1 is the minneighbour and 4
 is theminsize of the image

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,0),2)

    cv2.imshow("Video", frame)

    if cv2.waitKey(1) == ord('q'): #This adds a Delay and looks for the key press inorder to break the loop
        break
```

Question-3

Release the resources after recording and destroy all windows

In [ ]:
```
cap.release() #Release the resources after recording
cv2.destroyAllWindows()
```

If you wish to know how OpenCV works in Colab, Please click the link below,

https://colab.research.google.com/drive/1DBchZrcVIl_tGyLRfKXT9IzATe2rQ4js?usp=sharing
(https://colab.research.google.com/drive/1DBchZrcVIl_tGyLRfKXT9IzATe2rQ4js?usp=sharing)

Generative Adversarial Networks (GANs) are one of the most interesting ideas in computer science today. Two models are trained simultaneously by an adversarial process. A generator ("the artist") learns to create images that look real, while a discriminator ("the art critic") learns to tell real images apart from fakes

Types of GANs include **DCGAN**, **Conditional GANs**, **InfoGANs**, and **StackGANs**

DCGAN models are more stable and produce higher quality images

▪ It consists of two networks: **Discriminator** and **Generator**

▪ The Discriminator is made up of convolutional layers, batch norm layers, and LeakyRelu activations

▪ The Generator is comprised of convolutional transpose layers, batch norm layers, and ReLu activations

▪ Uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively

Scenario 1 : Implementing DCGAN in Tensorflow 2.x

Problem Statement

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

As a Deep Learning Engineer, your goal is to build a **Deep Convolutional Generative Adversarial Network** (DCGAN) to create images resembling Fashion MNIST Dataset in Tensorflow 2.x

Tasks to be performed

In this tutorial you will be performing the following tasks:

- Import Tensorflow and other required libraries - Beginner
- Load & pre-process the dataset for the model - Beginner
- Define the Generator Model - Advance
- Define the Discriminator i.e, a CNN-based Image Classifier - Advance
- Define the Training Loop - Advance
- Train the Model - Beginner
- Visualize the Final Results - Beginner

Dataset Description

The [Fashion MNIST Dataset (https://www.tensorflow.org/datasets/catalog/fashion_mnist)](https://www.tensorflow.org/datasets/catalog/fashion_mnist) is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

Each training and test example is assigned to one of the following labels:

Label Description

0 - T-shirt/top

1 - Trouser

2 - Pullover

3 - Dress

4 - Coat

5 - Sandal

6 - Shirt

7 - Sneaker

8 - Bag

9 - Ankle boot

Note : This Notebook will take almost 11 hours to run

Question 1:

Import Tensorflow and other required libraries

In [1]:

```
import glob
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
from tensorflow.keras import layers
import time
import tensorflow as tf
from IPython import display
```

Question 2 :

Load and Pre-process the Dataset for the Model

In [2]:

```
# Loading the Dataset
(train_images, train_labels), (_, _) = tf.keras.datasets.fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/train-labels-idx1-ubyte.gz
32768/29515 [=================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
```

We will be using the Fashion MNIST dataset to train the generator and the discriminator. The generator will generate Fashionable clothing item images resembling the Fashion MNIST data.

In [3]:

```
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).astype('float32')
train_images = (train_images - 127.5) / 127.5 # Normalize the images to [-1, 1]
```

In [4]:

```
BUFFER_SIZE = 60000
BATCH_SIZE = 300 # Batch size is the number of samples processed before the model is up
dated
```

In [5]:

```
# Batch and shuffle the data
# We can get the slices of an array in the form of objects by using tf.data.Dataset.fro
m_tensor_slices() method

train_dataset = tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).b
atch(BATCH_SIZE)
```

In [6]:

```
train_dataset
```

Out[6]:

```
<BatchDataset shapes: (None, 28, 28, 1), types: tf.float32>
```

Question 3:

Define the Generator Model

In [7]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D, Acti
vation,BatchNormalization,LeakyReLU,Conv2DTranspose,Reshape
def generator_model():
    model = Sequential()
    model.add(Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(BatchNormalization())
    model.add(LeakyReLU())

    model.add(Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256) # Note: None is the batch size

    model.add(Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_bias=Fal
se))
    #128 is the dimensionality of the output space
    #(5,5) specifies the height and width of the 2D convolution window
    assert model.output_shape == (None, 7, 7, 128)
    model.add(BatchNormalization())
    model.add(LeakyReLU())

    model.add(Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_bias=Fals
e))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(BatchNormalization())
    model.add(LeakyReLU())

    model.add(Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use_bias=False
, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)

    return model
```

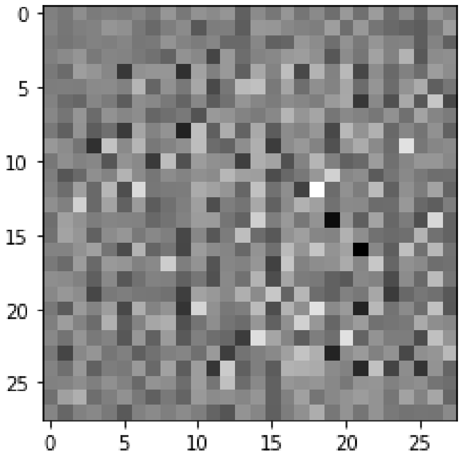
From above, you can see that the generator uses **tf.keras.layers.Conv2DTranspose** (upsampling) layers to produce an image from a seed (random noise). The **Dense** layer that takes this seed as input, then upsample several times until we reach the desired image size of **28x28x1**

In [8]:

```
# Let's use the Untrained Generator to create an Image

generator = generator_model()

noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)

plt.imshow(generated_image[0, :, :, 0], cmap='gray')
```

Out[8]:

```
<matplotlib.image.AxesImage at 0x7fcd68e55860>
```



Question 4 :

Define the Discriminator i.e, a CNN-based Image Classifier

In [9]:

```
def discriminator_model():
    model = Sequential()
    model.add(Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                                     input_shape=[28, 28, 1]))
    model.add(LeakyReLU())
    model.add(Dropout(0.3)) # to deactivate 30% neurons randomly to prevent overfitting

    model.add(Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(LeakyReLU())
    model.add(Dropout(0.3)) # to deactivate 30% neurons randomly to prevent overfitting

    model.add(Flatten()) # Converts multi dimensional array to 1D channel
    model.add(Dense(1))

    return model
```

In [10]:

```
#Use the Untrained Discriminator to classify the generated images as real or fake
#The model will be trained to output positive values for real images, and negative valu
es for fake images

discriminator = discriminator_model()
decision = discriminator(generated_image)
print (decision)
```

```
tf.Tensor([[0.00252824]], shape=(1, 1), dtype=float32)
```

If you want to learn more about the architecture for stable DCGANs **Click Here!** (https://arxiv.org/pdf/1511.06434)

Question 5 :

Define loss functions and optimizers for both models

In [11]:

```
from tensorflow.keras.losses import BinaryCrossentropy
# This method returns a helper function to compute cross entropy loss
cross_entropy = BinaryCrossentropy(from_logits=True)
```

In [12]:

```
# Discriminator Loss
# This method helps to distinguish between real images from fakes by discriminator
#Compares the discriminator's predictions on real images to an array of 1s,
#and the discriminator's predictions on fake (generated) images to an array of 0s

def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss
```

In [13]:

```
# The generator's loss quantifies how well it was able to fool the discriminator
# If the generator is performing well, the discriminator will classify the fake images
 as real (or 1)
# Here, we will compare the discriminators decisions on the generated images to an arra
y of 1s


def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)
```

In [14]:

```
# The Discriminator and the Generator Optimizers are different because we are training
 two networks separately

from tensorflow.keras.optimizers import Adam

generator_optimizer = Adam(1e-4)
discriminator_optimizer = Adam(1e-4)
```

Question 6 :

Define the Training Loop

The Training Loop begins with the

- Generator receiving a random seed as input.
- That seed is used to produce an image.
- The discriminator is then used to classify real images (drawn from the training set) and fakes images (produced by the generator).
- The loss is calculated for each of these models, and the gradients are used to update the generator and discriminator.

In [15]:

```
EPOCHS = 50 # Number of times that the model is exposed to the training dataset
noise_dim = 100
num_examples_to_generate = 16

# We will reuse this seed overtime
seed = tf.random.normal([num_examples_to_generate, noise_dim])
```

From above, you can see that the number of Epochs has been set to 50 which can be increased to improve the accuracy

Question 7 :

Train the model

In [16]:

```
# Notice the use of `tf.function`
# This annotation causes the function to be "compiled"

@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
      generated_images = generator(noise, training=True)

      real_output = discriminator(images, training=True)
      fake_output = discriminator(generated_images, training=True)

      gen_loss = generator_loss(fake_output)
      disc_loss = discriminator_loss(real_output, fake_output)

    gradients_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(disc_loss, discriminator.trainable_
variables)

    generator_optimizer.apply_gradients(zip(gradients_of_generator, generator.trainable
_variables))
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator, discriminat
or.trainable_variables))
```

In [17]:

```
def train(dataset, epochs):
  for epoch in range(epochs):
    start = time.time()

    for image_batch in dataset:
      train_step(image_batch)


    display.clear_output(wait=True)
    generate_and_save_images(generator,
                             epoch + 1,
                             seed)


    print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-start))

  # Generate after the final epoch
  display.clear_output(wait=True)
  generate_and_save_images(generator,
                           epochs,
                           seed)
```

In [18]:

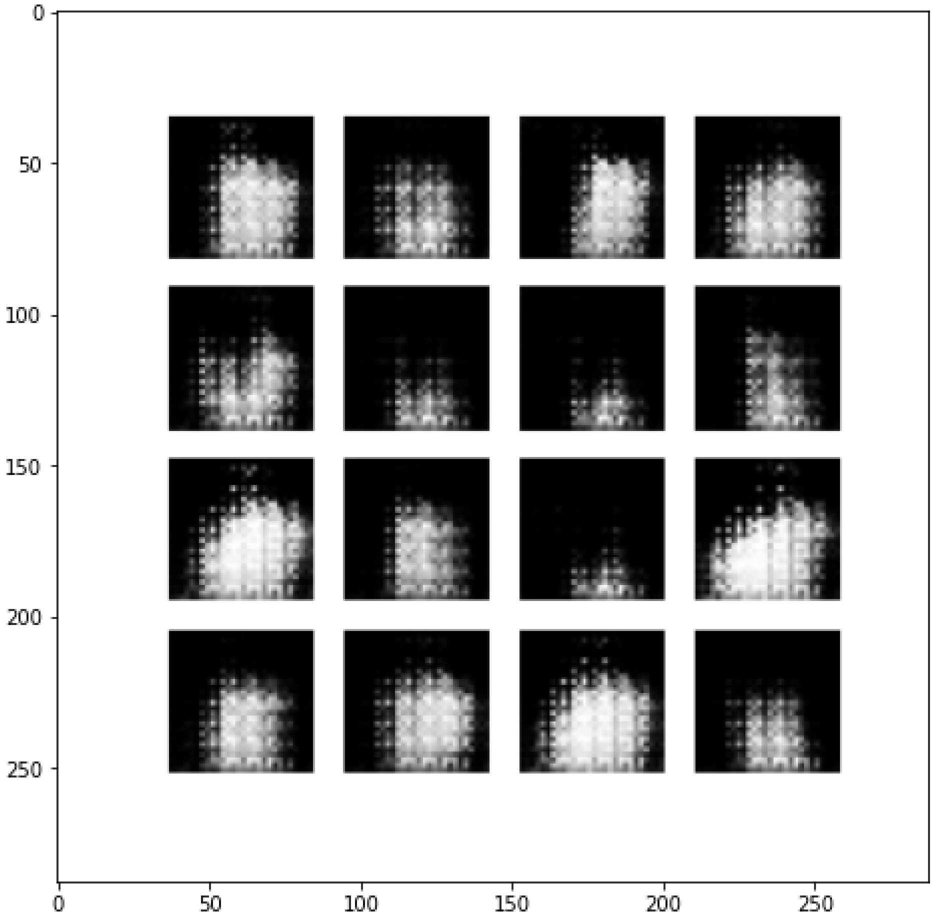
```
def generate_and_save_images(model, epoch, test_input):
  # Notice `training` is set to False
  # This is so all layers run in inference mode (batchnorm)
  predictions = model(test_input, training=False)

  fig = plt.figure(figsize=(4,4))

  for i in range(predictions.shape[0]):
      plt.subplot(4, 4, i+1)
      plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
      plt.axis('off')

  plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
  plt.show()
```

In [22]:

```
#Here, we are calling the train() method defined above to train the generator and discr
iminator simultaneously

train(train_dataset, 3) #Training the Model
```

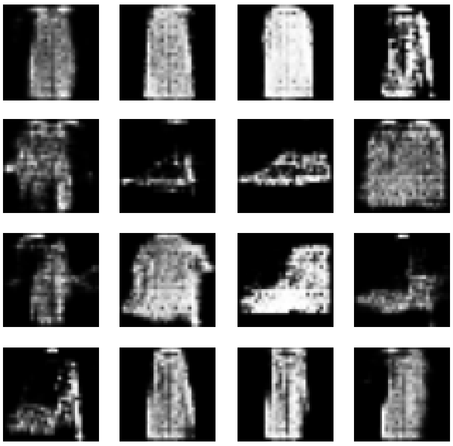
In [21]:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
plt.figure(figsize=(8,8))
img=mpimg.imread('/content/image_at_epoch_0003.png') # imread() function is used to rea
d image data in an ndarray object of float32 dtype
imgplot = plt.imshow(img)
plt.show()
```


0
50
100
150
200
250
0
50
100
150
200
250

In [ ]:

```
#Here, we are calling the train() method defined above to train the generator and discr
iminator simultaneously

train(train_dataset, EPOCHS) #Training the Model
```



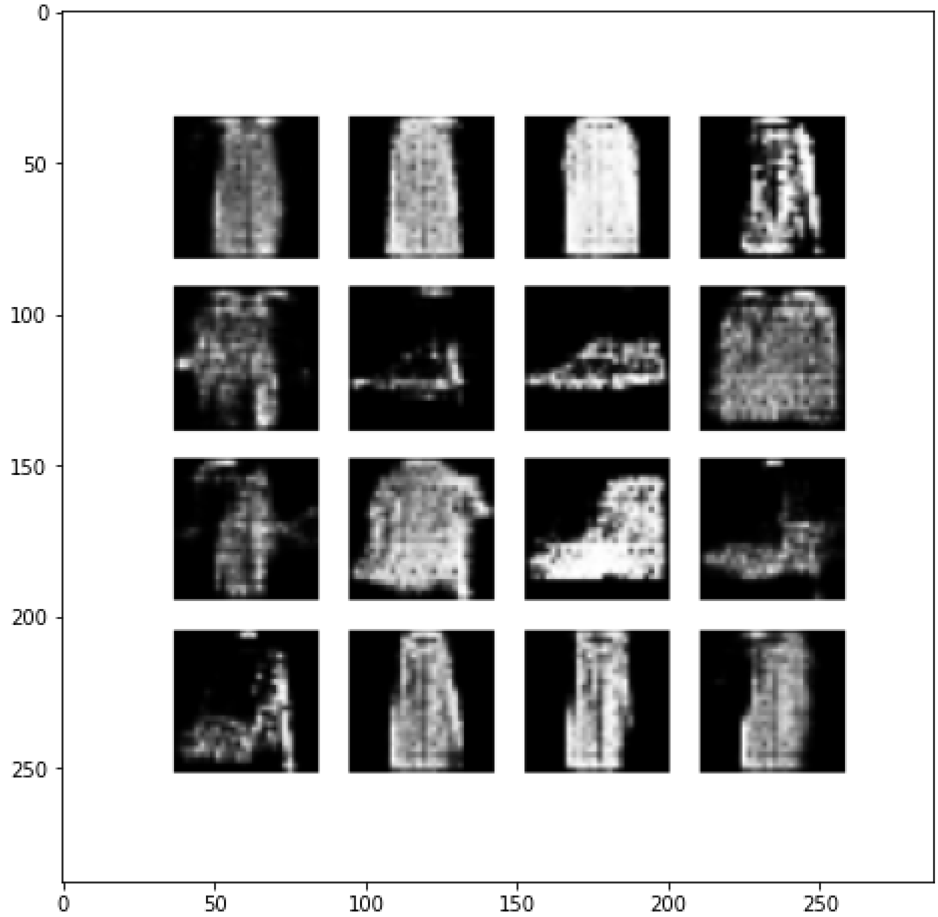
Note : At the beginning of the training, the generated images look like random noise. As training progresses, the generated images will look increasingly real. After about 50 epochs, they resemble Fashion MNIST clothing images.

Question 8 :

Visualize the Final Results

In [ ]:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
plt.figure(figsize=(8,8))
img=mpimg.imread('/content/image_at_epoch_0050.png') # imread() function is used to rea
d image data in an ndarray object of float32 dtype
imgplot = plt.imshow(img)
plt.show()
```



#Scenario 2: Traffic Surveillance

Concerned with the increasing number of road accidents the government of Bihar wants to create a smart surveillance system for traffic management.

###**Problem Statement:**

You as a machine learning engineer is told to create a model to detect objects on the road using the pre-trained YOLO model

###Tasks to be performed:

Load the YOLO model using Darknet repository - Beginner
Load the pre-trained weights of the YOLO-v3 model - Beginner
Generate and Display inferences on videos using pre-trained model - Intermediate

###Topics Covered:

Object Detection on video using pre-trained YOLO-v3
Transfer Learning

###Question-1: Download the source code for the YOLO model using darknet

In [ ]:
```
#clone darknet repository
import os
os.environ['PATH'] += ':/usr/local/cuda/bin'

!rm -fr darknet
!git clone https://github.com/AlexeyAB/darknet/
```

```
Cloning into 'darknet'...
remote: Enumerating objects: 13738, done.
remote: Total 13738 (delta 0), reused 0 (delta 0), pack-reused 137
38
Receiving objects: 100% (13738/13738), 12.30 MiB | 18.05 MiB/s, do
ne.
Resolving deltas: 100% (9372/9372), done.
```

In [ ]:

```
!apt install gcc-5 g++-5 -y
!ln -s /usr/bin/gcc-5 /usr/local/cuda/bin/gcc
!ln -s /usr/bin/g++-5 /usr/local/cuda/bin/g++
```

```
Reading package lists... Done

Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer
required:
  libnvidia-common-440
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  cpp-5 gcc-5-base libasan2 libgcc-5-dev libisl15 libmpx0 libstdc+
+-5-dev
Suggested packages:
  gcc-5-locales g++-5-multilib gcc-5-doc libstdc++6-5-dbg gcc-5-mu
ltilib
  libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan2-dbg
  liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx0-
dbg
  libquadmath0-dbg libstdc++-5-doc
The following NEW packages will be installed:
  cpp-5 g++-5 gcc-5 gcc-5-base libasan2 libgcc-5-dev libisl15 libm
px0
  libstdc++-5-dev
0 upgraded, 9 newly installed, 0 to remove and 59 not upgraded.
Need to get 29.1 MB of archives.
After this operation, 100 MB of additional disk space will be used
.
Get:1 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 gcc-5-bas
e amd64 5.5.0-12ubuntu1 [17.1 kB]
Get:2 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 libisl15
amd64 0.18-4 [548 kB]
Get:3 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 cpp-5 amd
64 5.5.0-12ubuntu1 [7,785 kB]
Get:4 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 libasan2
amd64 5.5.0-12ubuntu1 [264 kB]
Get:5 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 libmpx0 a
md64 5.5.0-12ubuntu1 [9,888 B]
Get:6 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 libgcc-5-
dev amd64 5.5.0-12ubuntu1 [2,224 kB]
Get:7 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 gcc-5 amd
64 5.5.0-12ubuntu1 [8,357 kB]
Get:8 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 libstdc++
-5-dev amd64 5.5.0-12ubuntu1 [1,415 kB]
```

```
Get:9 http://archive.ubuntu.com/ubuntu
(http://archive.ubuntu.com/ubuntu) bionic/universe amd64 g++-5 amd
64 5.5.0-12ubuntu1 [8,450 kB]
Fetched 29.1 MB in 2s (15.1 MB/s)
Selecting previously unselected package gcc-5-base:amd64.

(Reading database ... 144328 files and directories currently insta
lled.)
Preparing to unpack .../0-gcc-5-base_5.5.0-12ubuntu1_amd64.deb ...
Unpacking gcc-5-base:amd64 (5.5.0-12ubuntu1) ...
Selecting previously unselected package libisl15:amd64.
Preparing to unpack .../1-libisl15_0.18-4_amd64.deb ...
Unpacking libisl15:amd64 (0.18-4) ...
Selecting previously unselected package cpp-5.
Preparing to unpack .../2-cpp-5_5.5.0-12ubuntu1_amd64.deb ...
Unpacking cpp-5 (5.5.0-12ubuntu1) ...
Selecting previously unselected package libasan2:amd64.
Preparing to unpack .../3-libasan2_5.5.0-12ubuntu1_amd64.deb ...
Unpacking libasan2:amd64 (5.5.0-12ubuntu1) ...
Selecting previously unselected package libmpx0:amd64.
Preparing to unpack .../4-libmpx0_5.5.0-12ubuntu1_amd64.deb ...
Unpacking libmpx0:amd64 (5.5.0-12ubuntu1) ...
Selecting previously unselected package libgcc-5-dev:amd64.
Preparing to unpack .../5-libgcc-5-dev_5.5.0-12ubuntu1_amd64.deb .
..
Unpacking libgcc-5-dev:amd64 (5.5.0-12ubuntu1) ...
Selecting previously unselected package gcc-5.
Preparing to unpack .../6-gcc-5_5.5.0-12ubuntu1_amd64.deb ...
Unpacking gcc-5 (5.5.0-12ubuntu1) ...
Selecting previously unselected package libstdc++-5-dev:amd64.
Preparing to unpack .../7-libstdc++-5-dev_5.5.0-12ubuntu1_amd64.de
b ...
Unpacking libstdc++-5-dev:amd64 (5.5.0-12ubuntu1) ...
Selecting previously unselected package g++-5.
Preparing to unpack .../8-g++-5_5.5.0-12ubuntu1_amd64.deb ...
Unpacking g++-5 (5.5.0-12ubuntu1) ...
Setting up libisl15:amd64 (0.18-4) ...
Setting up gcc-5-base:amd64 (5.5.0-12ubuntu1) ...
Setting up libmpx0:amd64 (5.5.0-12ubuntu1) ...
Setting up libasan2:amd64 (5.5.0-12ubuntu1) ...
Setting up libgcc-5-dev:amd64 (5.5.0-12ubuntu1) ...
Setting up cpp-5 (5.5.0-12ubuntu1) ...
Setting up libstdc++-5-dev:amd64 (5.5.0-12ubuntu1) ...
Setting up gcc-5 (5.5.0-12ubuntu1) ...
Setting up g++-5 (5.5.0-12ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
/sbin/ldconfig.real: /usr/local/lib/python3.6/dist-packages/ideep4
py/lib/libmkldnn.so.0 is not a symbolic link
```

###Question-2: Enable GPU and OpenCV support from the darknet, and compile the model

###Question-3: Download the pre-trained weights of YOLO-v3

In [ ]:
```
%cd darknet
!sed -i 's/GPU=0/GPU=1/g' Makefile
!sed -i 's/OPENCV=0/OPENCV=1/g' Makefile
!make
```

In [ ]:
```
# get yolov3 weights
!wget https://pjreddie.com/media/files/yolov3.weights
```

```
--2020-06-25 07:49:49--  https://pjreddie.com/media/files/yolov3.w
eights (https://pjreddie.com/media/files/yolov3.weights)
Resolving pjreddie.com (pjreddie.com)... 128.208.4.108
Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: 248007048 (237M) [application/octet-stream]
Saving to: ‘yolov3.weights’

yolov3.weights      100%[===================>] 236.52M   367KB/s
in 7m 25s

2020-06-25 07:57:16 (544 KB/s) - ‘yolov3.weights’ saved [248007048
/248007048]
```

###Question-4: Make the darknet executable, also print the current working directory

In [ ]:
```
!chmod a+x ./darknet
!pwd
```

```
/content/darknet
```

download the video

In [ ]:
```
!wget https://www.dropbox.com/s/7ppejm1c0uzezt1/P1033673.mp4
```

```
--2020-06-25 07:57:22--  https://www.dropbox.com/s/7ppejm1c0uzezt1
/P1033673.mp4
(https://www.dropbox.com/s/7ppejm1c0uzezt1/P1033673.mp4)
Resolving www.dropbox.com (www.dropbox.com)... 162.125.3.1, 2620:1
00:6018:1::a27d:301
Connecting to www.dropbox.com (www.dropbox.com)|162.125.3.1|:443..
. connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/7ppejm1c0uzezt1/P1033673.mp4 [following]
--2020-06-25 07:57:22--  https://www.dropbox.com/s/raw/7ppejm1c0uz
ezt1/P1033673.mp4
(https://www.dropbox.com/s/raw/7ppejm1c0uzezt1/P1033673.mp4)
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc45170b5407f4469ff2f5628600.dl.dropboxuserconte
nt.com/cd/0/inline/A6Q2QDxUvQF_WD1iXjr66MEaQE9rrhESt5sTamN3HcgnOgk
D-Rn4DfC8VDgj87JytYWmbuwN8EdwG75OIr5rGt5MezQ3yAQm0w39QGdxlg_qLlrzz
Denpcl753lJwjErD64/file#
(https://uc45170b5407f4469ff2f5628600.dl.dropboxusercontent.com/cd
/0/inline/A6Q2QDxUvQF_WD1iXjr66MEaQE9rrhESt5sTamN3HcgnOgkD-Rn4DfC8
VDgj87JytYWmbuwN8EdwG75OIr5rGt5MezQ3yAQm0w39QGdxlg_qLlrzzDenpcl753
lJwjErD64/file#) [following]
--2020-06-25 07:57:23--  https://uc45170b5407f4469ff2f5628600.dl.d
ropboxusercontent.com/cd/0/inline/A6Q2QDxUvQF_WD1iXjr66MEaQE9rrhES
t5sTamN3HcgnOgkD-Rn4DfC8VDgj87JytYWmbuwN8EdwG75OIr5rGt5MezQ3yAQm0w
39QGdxlg_qLlrzzDenpcl753lJwjErD64/file
(https://uc45170b5407f4469ff2f5628600.dl.dropboxusercontent.com/cd
/0/inline/A6Q2QDxUvQF_WD1iXjr66MEaQE9rrhESt5sTamN3HcgnOgkD-Rn4DfC8
VDgj87JytYWmbuwN8EdwG75OIr5rGt5MezQ3yAQm0w39QGdxlg_qLlrzzDenpcl753
lJwjErD64/file)
Resolving uc45170b5407f4469ff2f5628600.dl.dropboxusercontent.com (
uc45170b5407f4469ff2f5628600.dl.dropboxusercontent.com)... 162.125
.3.15, 2620:100:6018:15::a27d:30f
Connecting to uc45170b5407f4469ff2f5628600.dl.dropboxusercontent.c
om (uc45170b5407f4469ff2f5628600.dl.dropboxusercontent.com)|162.12
5.3.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 190850768 (182M) [video/mp4]
Saving to: ‘P1033673.mp4’

P1033673.mp4        100%[===================>] 182.01M  44.7MB/s
in 4.0s

2020-06-25 07:57:27 (45.7 MB/s) - ‘P1033673.mp4’ saved [190850768/
190850768]
```

###Question-5: Perform prediction on the video using yolo-v3 and save the output in a video file

In [ ]:
```
!./darknet detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weight
```

```
Streaming output truncated to the last 5000 lines.
car: 98%
car: 97%
car: 97%
car: 96%
car: 95%
car: 95%
car: 94%
car: 93%
car: 93%
car: 84%
car: 80%
car: 79%
car: 71%
person: 99%
person: 90%
person: 73%

FPS:4.4          AVG_FPS:4.2
```

Download the video 'output.avi' and check the output. In the video you can see how the model is able to detect multiple objects (percon, car, etc.)

In [ ]: