Uploading the File

First we upload the zip file and later we unzip the file

source: https://www.kaggle.com/datasets/otahharrison/ml-image-classification

(Note: Few of the code snippets used in this notebook is taken from Professor's notebooks on github repo)

In [50]:
```
from google.colab import files
uploaded = files.upload()
```

Browse... No files selected. Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving ML_image_classification.zip to ML_image_classification (1).zip
```

In [51]:
```
!unzip ML_image_classification.zip
```

```
Archive:  ML_image_classification.zip
replace DATA_IMAGE/Test/CAR/1.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: DATA_IMAGE/Test/CAR/1.jpg
replace DATA_IMAGE/Test/CAR/10.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: a
error:  invalid response [a]
replace DATA_IMAGE/Test/CAR/10.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: a
error:  invalid response [a]
replace DATA_IMAGE/Test/CAR/10.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: DATA_IMAGE/Test/CAR/10.jpg
  inflating: DATA_IMAGE/Test/CAR/100.jpg
  inflating: DATA_IMAGE/Test/CAR/11.jpg
  inflating: DATA_IMAGE/Test/CAR/12.jpg
  inflating: DATA_IMAGE/Test/CAR/13.jpg
  inflating: DATA_IMAGE/Test/CAR/14.jpg
  inflating: DATA_IMAGE/Test/CAR/15.jpg
  inflating: DATA_IMAGE/Test/CAR/16.jpg
  inflating: DATA_IMAGE/Test/CAR/17.jpg
  inflating: DATA_IMAGE/Test/CAR/18.jpg
  inflating: DATA_IMAGE/Test/CAR/19.jpg
  inflating: DATA_IMAGE/Test/CAR/2.jpg
  inflating: DATA_IMAGE/Test/CAR/20.jpg
  inflating: DATA_IMAGE/Test/CAR/21.jpg
  inflating: DATA_IMAGE/Test/CAR/22.jpg
  inflating: DATA_IMAGE/Test/CAR/23.jpg
  inflating: DATA_IMAGE/Test/CAR/24.jpg
  inflating: DATA_IMAGE/Test/CAR/25.jpg
  inflating: DATA_IMAGE/Test/CAR/26.jpg
  inflating: DATA_IMAGE/Test/CAR/27.jpg
  inflating: DATA_IMAGE/Test/CAR/28.jpg
  inflating: DATA_IMAGE/Test/CAR/29.jpg
  inflating: DATA_IMAGE/Test/CAR/3.jpg
  inflating: DATA_IMAGE/Test/CAR/30.jpg
  inflating: DATA_IMAGE/Test/CAR/31.jpg
  inflating: DATA_IMAGE/Test/CAR/32.jpg
  inflating: DATA_IMAGE/Test/CAR/33.jpg
  inflating: DATA_IMAGE/Test/CAR/34.jpg
  inflating: DATA_IMAGE/Test/CAR/35.jpg
  inflating: DATA_IMAGE/Test/CAR/36.jpg
  inflating: DATA_IMAGE/Test/CAR/37.jpg
  inflating: DATA_IMAGE/Test/CAR/38.jpg
  inflating: DATA_IMAGE/Test/CAR/39.jpg
  inflating: DATA_IMAGE/Test/CAR/4.jpg
  inflating: DATA_IMAGE/Test/CAR/40.jpg
  inflating: DATA_IMAGE/Test/CAR/41.jpg
  inflating: DATA_IMAGE/Test/CAR/42.jpg
  inflating: DATA_IMAGE/Test/CAR/43.jpg
  inflating: DATA_IMAGE/Test/CAR/44.jpg
  inflating: DATA_IMAGE/Test/CAR/45.jpg
  inflating: DATA_IMAGE/Test/CAR/46.jpg
  inflating: DATA_IMAGE/Test/CAR/47.jpg
  inflating: DATA_IMAGE/Test/CAR/48.jpg
  inflating: DATA_IMAGE/Test/CAR/49.jpg
  inflating: DATA_IMAGE/Test/CAR/5.jpg
  inflating: DATA_IMAGE/Test/CAR/50.jpg
  inflating: DATA_IMAGE/Test/CAR/51.jpg
  inflating: DATA_IMAGE/Test/CAR/52.jpg
  inflating: DATA_IMAGE/Test/CAR/53.jpg
  inflating: DATA_IMAGE/Test/CAR/54.jpg
  inflating: DATA_IMAGE/Test/CAR/55.jpg
```

```
inflating: DATA_IMAGE/Test/CAR/56.jpg
inflating: DATA_IMAGE/Test/CAR/57.jpg
inflating: DATA_IMAGE/Test/CAR/58.jpg
inflating: DATA_IMAGE/Test/CAR/59.jpg
inflating: DATA_IMAGE/Test/CAR/6.jpg
inflating: DATA_IMAGE/Test/CAR/60.jpg
inflating: DATA_IMAGE/Test/CAR/61.jpg
inflating: DATA_IMAGE/Test/CAR/62.jpg
inflating: DATA_IMAGE/Test/CAR/63.jpg
inflating: DATA_IMAGE/Test/CAR/64.jpg
inflating: DATA_IMAGE/Test/CAR/65.jpg
inflating: DATA_IMAGE/Test/CAR/66.jpg
inflating: DATA_IMAGE/Test/CAR/67.jpg
inflating: DATA_IMAGE/Test/CAR/68.jpg
inflating: DATA_IMAGE/Test/CAR/69.jpg
inflating: DATA_IMAGE/Test/CAR/7.jpg
inflating: DATA_IMAGE/Test/CAR/70.jpg
inflating: DATA_IMAGE/Test/CAR/71.jpg
inflating: DATA_IMAGE/Test/CAR/72.jpg
inflating: DATA_IMAGE/Test/CAR/73.jpg
inflating: DATA_IMAGE/Test/CAR/74.jpg
inflating: DATA_IMAGE/Test/CAR/75.jpg
inflating: DATA_IMAGE/Test/CAR/76.jpg
inflating: DATA_IMAGE/Test/CAR/77.jpg
inflating: DATA_IMAGE/Test/CAR/78.jpg
inflating: DATA_IMAGE/Test/CAR/79.jpg
inflating: DATA_IMAGE/Test/CAR/8.jpg
inflating: DATA_IMAGE/Test/CAR/80.jpg
inflating: DATA_IMAGE/Test/CAR/81.jpg
inflating: DATA_IMAGE/Test/CAR/82.jpg
inflating: DATA_IMAGE/Test/CAR/83.jpg
inflating: DATA_IMAGE/Test/CAR/84.jpg
inflating: DATA_IMAGE/Test/CAR/85.jpg
inflating: DATA_IMAGE/Test/CAR/86.jpg
inflating: DATA_IMAGE/Test/CAR/87.jpg
inflating: DATA_IMAGE/Test/CAR/88.jpg
inflating: DATA_IMAGE/Test/CAR/89.jpg
inflating: DATA_IMAGE/Test/CAR/9.jpg
inflating: DATA_IMAGE/Test/CAR/90.jpg
inflating: DATA_IMAGE/Test/CAR/91.jpg
inflating: DATA_IMAGE/Test/CAR/92.jpg
inflating: DATA_IMAGE/Test/CAR/93.jpg
inflating: DATA_IMAGE/Test/CAR/94.jpg
inflating: DATA_IMAGE/Test/CAR/95.jpg
inflating: DATA_IMAGE/Test/CAR/96.jpg
inflating: DATA_IMAGE/Test/CAR/97.jpg
inflating: DATA_IMAGE/Test/CAR/98.jpg
inflating: DATA_IMAGE/Test/CAR/99.jpg
inflating: DATA_IMAGE/Test/TREE/096.jpg
inflating: DATA_IMAGE/Test/TREE/10.jpg
inflating: DATA_IMAGE/Test/TREE/100.jpg
inflating: DATA_IMAGE/Test/TREE/101.jpg
inflating: DATA_IMAGE/Test/TREE/102.jpg
inflating: DATA_IMAGE/Test/TREE/103.jpg
inflating: DATA_IMAGE/Test/TREE/11.jpg
inflating: DATA_IMAGE/Test/TREE/12.jpeg
inflating: DATA_IMAGE/Test/TREE/13.jpg
inflating: DATA_IMAGE/Test/TREE/14.jpg
inflating: DATA_IMAGE/Test/TREE/15.jpg
```

```
inflating: DATA_IMAGE/Test/TREE/16.jpeg
inflating: DATA_IMAGE/Test/TREE/17.jpg
inflating: DATA_IMAGE/Test/TREE/18.jpg
inflating: DATA_IMAGE/Test/TREE/19.jpg
inflating: DATA_IMAGE/Test/TREE/20.jpg
inflating: DATA_IMAGE/Test/TREE/21.jpg
inflating: DATA_IMAGE/Test/TREE/22.jpg
inflating: DATA_IMAGE/Test/TREE/23.jpg
inflating: DATA_IMAGE/Test/TREE/24.jpg
inflating: DATA_IMAGE/Test/TREE/25.jpg
inflating: DATA_IMAGE/Test/TREE/26.jpg
inflating: DATA_IMAGE/Test/TREE/27.jpg
inflating: DATA_IMAGE/Test/TREE/28.jpg
inflating: DATA_IMAGE/Test/TREE/29.jpeg
inflating: DATA_IMAGE/Test/TREE/30.jpeg
inflating: DATA_IMAGE/Test/TREE/31.jpeg
inflating: DATA_IMAGE/Test/TREE/32.jpg
inflating: DATA_IMAGE/Test/TREE/33.jpg
inflating: DATA_IMAGE/Test/TREE/34.jpg
inflating: DATA_IMAGE/Test/TREE/35.jpg
inflating: DATA_IMAGE/Test/TREE/36.jpg
inflating: DATA_IMAGE/Test/TREE/37.jpg
inflating: DATA_IMAGE/Test/TREE/38.jpg
inflating: DATA_IMAGE/Test/TREE/39.jpg
inflating: DATA_IMAGE/Test/TREE/4.jpg
inflating: DATA_IMAGE/Test/TREE/40.jpeg
inflating: DATA_IMAGE/Test/TREE/41.jpeg
inflating: DATA_IMAGE/Test/TREE/42.jpeg
inflating: DATA_IMAGE/Test/TREE/43.jpeg
inflating: DATA_IMAGE/Test/TREE/44.jpeg
inflating: DATA_IMAGE/Test/TREE/45.jpeg
inflating: DATA_IMAGE/Test/TREE/46.jpg
inflating: DATA_IMAGE/Test/TREE/47.jpeg
inflating: DATA_IMAGE/Test/TREE/48.jpeg
inflating: DATA_IMAGE/Test/TREE/49.jpg
inflating: DATA_IMAGE/Test/TREE/5.jpg
inflating: DATA_IMAGE/Test/TREE/50.jpg
inflating: DATA_IMAGE/Test/TREE/51.jpeg
inflating: DATA_IMAGE/Test/TREE/52.jpg
inflating: DATA_IMAGE/Test/TREE/53.jpg
inflating: DATA_IMAGE/Test/TREE/54.jpg
inflating: DATA_IMAGE/Test/TREE/55.jpeg
inflating: DATA_IMAGE/Test/TREE/56.jpg
inflating: DATA_IMAGE/Test/TREE/57.jpeg
inflating: DATA_IMAGE/Test/TREE/58.jpg
inflating: DATA_IMAGE/Test/TREE/59.jpeg
inflating: DATA_IMAGE/Test/TREE/6.jpeg
inflating: DATA_IMAGE/Test/TREE/60.jpeg
inflating: DATA_IMAGE/Test/TREE/61.jpeg
inflating: DATA_IMAGE/Test/TREE/62.jpg
inflating: DATA_IMAGE/Test/TREE/63.jpg
inflating: DATA_IMAGE/Test/TREE/64.jpg
inflating: DATA_IMAGE/Test/TREE/65.jpeg
inflating: DATA_IMAGE/Test/TREE/66.jpeg
inflating: DATA_IMAGE/Test/TREE/67.jpg
inflating: DATA_IMAGE/Test/TREE/68.jpeg
inflating: DATA_IMAGE/Test/TREE/69.jpg
inflating: DATA_IMAGE/Test/TREE/7.jpeg
inflating: DATA_IMAGE/Test/TREE/70.jpeg
```

```
inflating: DATA_IMAGE/Test/TREE/71.jpg
inflating: DATA_IMAGE/Test/TREE/72.jpeg
inflating: DATA_IMAGE/Test/TREE/73.jpg
inflating: DATA_IMAGE/Test/TREE/74.jpg
inflating: DATA_IMAGE/Test/TREE/75.jpeg
inflating: DATA_IMAGE/Test/TREE/76.jpeg
inflating: DATA_IMAGE/Test/TREE/77.jpeg
inflating: DATA_IMAGE/Test/TREE/78.jpg
inflating: DATA_IMAGE/Test/TREE/79.jpg
inflating: DATA_IMAGE/Test/TREE/8.jpg
inflating: DATA_IMAGE/Test/TREE/80.jpg
inflating: DATA_IMAGE/Test/TREE/81.jpg
inflating: DATA_IMAGE/Test/TREE/82.jpg
inflating: DATA_IMAGE/Test/TREE/83.jpg
inflating: DATA_IMAGE/Test/TREE/84.jpeg
inflating: DATA_IMAGE/Test/TREE/85.jpg
inflating: DATA_IMAGE/Test/TREE/86.jpg
inflating: DATA_IMAGE/Test/TREE/87.jpg
inflating: DATA_IMAGE/Test/TREE/88.jpg
inflating: DATA_IMAGE/Test/TREE/89.jpg
inflating: DATA_IMAGE/Test/TREE/9.jpg
inflating: DATA_IMAGE/Test/TREE/90.jpeg
inflating: DATA_IMAGE/Test/TREE/91.jpg
inflating: DATA_IMAGE/Test/TREE/92.jpg
inflating: DATA_IMAGE/Test/TREE/93.jpg
inflating: DATA_IMAGE/Test/TREE/94.jpeg
inflating: DATA_IMAGE/Test/TREE/95.jpg
inflating: DATA_IMAGE/Test/TREE/97.jpeg
inflating: DATA_IMAGE/Test/TREE/98.jpeg
inflating: DATA_IMAGE/Test/TREE/99.jpg
inflating: DATA_IMAGE/Train/CARS/101.jpg
inflating: DATA_IMAGE/Train/CARS/102.jpg
inflating: DATA_IMAGE/Train/CARS/103.jpg
inflating: DATA_IMAGE/Train/CARS/104.jpg
inflating: DATA_IMAGE/Train/CARS/105.jpg
inflating: DATA_IMAGE/Train/CARS/106.jpg
inflating: DATA_IMAGE/Train/CARS/107.jpg
inflating: DATA_IMAGE/Train/CARS/108.jpg
inflating: DATA_IMAGE/Train/CARS/109.jpg
inflating: DATA_IMAGE/Train/CARS/110.jpg
inflating: DATA_IMAGE/Train/CARS/111.jpg
inflating: DATA_IMAGE/Train/CARS/112.jpg
inflating: DATA_IMAGE/Train/CARS/113.jpg
inflating: DATA_IMAGE/Train/CARS/114.jpg
inflating: DATA_IMAGE/Train/CARS/115.jpg
inflating: DATA_IMAGE/Train/CARS/116.jpg
inflating: DATA_IMAGE/Train/CARS/117.jpg
inflating: DATA_IMAGE/Train/CARS/118.jpg
inflating: DATA_IMAGE/Train/CARS/119.jpg
inflating: DATA_IMAGE/Train/CARS/120.jpg
inflating: DATA_IMAGE/Train/CARS/121.jpg
inflating: DATA_IMAGE/Train/CARS/122.jpg
inflating: DATA_IMAGE/Train/CARS/123.jpg
inflating: DATA_IMAGE/Train/CARS/124.jpg
inflating: DATA_IMAGE/Train/CARS/125.jpg
inflating: DATA_IMAGE/Train/CARS/126.jpg
inflating: DATA_IMAGE/Train/CARS/127.jpg
inflating: DATA_IMAGE/Train/CARS/128.jpg
inflating: DATA_IMAGE/Train/CARS/129.jpg
```

```
inflating: DATA_IMAGE/Train/CARS/130.jpg
inflating: DATA_IMAGE/Train/CARS/131.jpg
inflating: DATA_IMAGE/Train/CARS/132.jpg
inflating: DATA_IMAGE/Train/CARS/133.jpg
inflating: DATA_IMAGE/Train/CARS/134.jpg
inflating: DATA_IMAGE/Train/CARS/135.jpg
inflating: DATA_IMAGE/Train/CARS/136.jpg
inflating: DATA_IMAGE/Train/CARS/137.jpg
inflating: DATA_IMAGE/Train/CARS/138.jpg
inflating: DATA_IMAGE/Train/CARS/139.jpg
inflating: DATA_IMAGE/Train/CARS/140.jpg
inflating: DATA_IMAGE/Train/CARS/141.jpg
inflating: DATA_IMAGE/Train/CARS/142.jpg
inflating: DATA_IMAGE/Train/CARS/143.jpg
inflating: DATA_IMAGE/Train/CARS/144.jpg
inflating: DATA_IMAGE/Train/CARS/145.jpg
inflating: DATA_IMAGE/Train/CARS/146.jpg
inflating: DATA_IMAGE/Train/CARS/147.jpg
inflating: DATA_IMAGE/Train/CARS/148.jpg
inflating: DATA_IMAGE/Train/CARS/149.jpg
inflating: DATA_IMAGE/Train/CARS/150.jpg
inflating: DATA_IMAGE/Train/CARS/151.jpg
inflating: DATA_IMAGE/Train/CARS/152.jpg
inflating: DATA_IMAGE/Train/CARS/153.jpg
inflating: DATA_IMAGE/Train/CARS/154.jpg
inflating: DATA_IMAGE/Train/CARS/155.jpg
inflating: DATA_IMAGE/Train/CARS/156.jpg
inflating: DATA_IMAGE/Train/CARS/157.jpg
inflating: DATA_IMAGE/Train/CARS/158.jpg
inflating: DATA_IMAGE/Train/CARS/159.jpg
inflating: DATA_IMAGE/Train/CARS/160.jpg
inflating: DATA_IMAGE/Train/CARS/161.jpg
inflating: DATA_IMAGE/Train/CARS/162.jpg
inflating: DATA_IMAGE/Train/CARS/163.jpg
inflating: DATA_IMAGE/Train/CARS/164.jpg
inflating: DATA_IMAGE/Train/CARS/165.jpg
inflating: DATA_IMAGE/Train/CARS/166.jpg
inflating: DATA_IMAGE/Train/CARS/167.jpg
inflating: DATA_IMAGE/Train/CARS/168.jpg
inflating: DATA_IMAGE/Train/CARS/169.jpg
inflating: DATA_IMAGE/Train/CARS/170.jpg
inflating: DATA_IMAGE/Train/CARS/171.jpg
inflating: DATA_IMAGE/Train/CARS/172.jpg
inflating: DATA_IMAGE/Train/CARS/173.jpg
inflating: DATA_IMAGE/Train/CARS/174.jpg
inflating: DATA_IMAGE/Train/CARS/175.jpg
inflating: DATA_IMAGE/Train/CARS/176.jpg
inflating: DATA_IMAGE/Train/CARS/177.jpg
inflating: DATA_IMAGE/Train/CARS/178.jpg
inflating: DATA_IMAGE/Train/CARS/179.jpg
inflating: DATA_IMAGE/Train/CARS/180.jpg
inflating: DATA_IMAGE/Train/CARS/181.jpg
inflating: DATA_IMAGE/Train/CARS/182.jpg
inflating: DATA_IMAGE/Train/CARS/183.jpg
inflating: DATA_IMAGE/Train/CARS/184.jpg
inflating: DATA_IMAGE/Train/CARS/185.jpg
inflating: DATA_IMAGE/Train/CARS/186.jpg
inflating: DATA_IMAGE/Train/CARS/187.jpg
inflating: DATA_IMAGE/Train/CARS/188.jpg
```

```
inflating: DATA_IMAGE/Train/CARS/189.jpg
inflating: DATA_IMAGE/Train/CARS/190.jpg
inflating: DATA_IMAGE/Train/CARS/191.jpg
inflating: DATA_IMAGE/Train/CARS/192.jpg
inflating: DATA_IMAGE/Train/CARS/193.jpg
inflating: DATA_IMAGE/Train/CARS/194.jpg
inflating: DATA_IMAGE/Train/CARS/195.jpg
inflating: DATA_IMAGE/Train/CARS/196.jpg
inflating: DATA_IMAGE/Train/CARS/197.jpg
inflating: DATA_IMAGE/Train/CARS/198.jpg
inflating: DATA_IMAGE/Train/CARS/199.jpg
inflating: DATA_IMAGE/Train/CARS/200.jpg
inflating: DATA_IMAGE/Train/TREE/1.jpg
inflating: DATA_IMAGE/Train/TREE/104.jpg
inflating: DATA_IMAGE/Train/TREE/105.jpg
inflating: DATA_IMAGE/Train/TREE/106.jpg
inflating: DATA_IMAGE/Train/TREE/107.jpg
inflating: DATA_IMAGE/Train/TREE/108.jpg
inflating: DATA_IMAGE/Train/TREE/109.jpg
inflating: DATA_IMAGE/Train/TREE/110.jpg
inflating: DATA_IMAGE/Train/TREE/111.jpg
inflating: DATA_IMAGE/Train/TREE/112.jpg
inflating: DATA_IMAGE/Train/TREE/113.jpg
inflating: DATA_IMAGE/Train/TREE/114.jpg
inflating: DATA_IMAGE/Train/TREE/115.jpg
inflating: DATA_IMAGE/Train/TREE/116.jpg
inflating: DATA_IMAGE/Train/TREE/117.jpg
inflating: DATA_IMAGE/Train/TREE/118.jpg
inflating: DATA_IMAGE/Train/TREE/119.jpg
inflating: DATA_IMAGE/Train/TREE/120.jpg
inflating: DATA_IMAGE/Train/TREE/121.jpeg
inflating: DATA_IMAGE/Train/TREE/122.jpg
inflating: DATA_IMAGE/Train/TREE/123.jpeg
inflating: DATA_IMAGE/Train/TREE/124.jpeg
inflating: DATA_IMAGE/Train/TREE/125.jpeg
inflating: DATA_IMAGE/Train/TREE/126.jpg
inflating: DATA_IMAGE/Train/TREE/127.jpeg
inflating: DATA_IMAGE/Train/TREE/128.jpg
inflating: DATA_IMAGE/Train/TREE/129.jpg
inflating: DATA_IMAGE/Train/TREE/130.jpg
inflating: DATA_IMAGE/Train/TREE/131.jpg
inflating: DATA_IMAGE/Train/TREE/132.jpg
inflating: DATA_IMAGE/Train/TREE/133.jpeg
inflating: DATA_IMAGE/Train/TREE/134.jpg
inflating: DATA_IMAGE/Train/TREE/135.jpeg
inflating: DATA_IMAGE/Train/TREE/135.jpg
inflating: DATA_IMAGE/Train/TREE/136.jpg
inflating: DATA_IMAGE/Train/TREE/137.jpg
inflating: DATA_IMAGE/Train/TREE/138.jpg
inflating: DATA_IMAGE/Train/TREE/139.jpeg
inflating: DATA_IMAGE/Train/TREE/140.jpg
inflating: DATA_IMAGE/Train/TREE/141.jpg
inflating: DATA_IMAGE/Train/TREE/142.jpg
inflating: DATA_IMAGE/Train/TREE/143.jpg
inflating: DATA_IMAGE/Train/TREE/144.jpg
inflating: DATA_IMAGE/Train/TREE/145.jpeg
inflating: DATA_IMAGE/Train/TREE/146.jpeg
inflating: DATA_IMAGE/Train/TREE/147.jpeg
inflating: DATA_IMAGE/Train/TREE/148.jpg
```

```
inflating: DATA_IMAGE/Train/TREE/149.jpeg
inflating: DATA_IMAGE/Train/TREE/150.jpeg
inflating: DATA_IMAGE/Train/TREE/151.jpg
inflating: DATA_IMAGE/Train/TREE/152.jpeg
inflating: DATA_IMAGE/Train/TREE/153.jpeg
inflating: DATA_IMAGE/Train/TREE/154.jpg
inflating: DATA_IMAGE/Train/TREE/155.jpeg
inflating: DATA_IMAGE/Train/TREE/156.jpg
inflating: DATA_IMAGE/Train/TREE/157.jpg
inflating: DATA_IMAGE/Train/TREE/158.jpeg
inflating: DATA_IMAGE/Train/TREE/159.jpg
inflating: DATA_IMAGE/Train/TREE/160.jpeg
inflating: DATA_IMAGE/Train/TREE/161.jpeg
inflating: DATA_IMAGE/Train/TREE/162.jpg
inflating: DATA_IMAGE/Train/TREE/163.jpg
inflating: DATA_IMAGE/Train/TREE/164.jpg
inflating: DATA_IMAGE/Train/TREE/165.jpg
inflating: DATA_IMAGE/Train/TREE/166.jpg
inflating: DATA_IMAGE/Train/TREE/167.jpg
inflating: DATA_IMAGE/Train/TREE/168.jpeg
inflating: DATA_IMAGE/Train/TREE/169.jpg
inflating: DATA_IMAGE/Train/TREE/170.jpg
inflating: DATA_IMAGE/Train/TREE/171.jpg
inflating: DATA_IMAGE/Train/TREE/172.jpg
inflating: DATA_IMAGE/Train/TREE/173.jpg
inflating: DATA_IMAGE/Train/TREE/174.jpg
inflating: DATA_IMAGE/Train/TREE/176.jpg
inflating: DATA_IMAGE/Train/TREE/177.jpg
inflating: DATA_IMAGE/Train/TREE/178.jpg
inflating: DATA_IMAGE/Train/TREE/179.jpeg
inflating: DATA_IMAGE/Train/TREE/180.jpg
inflating: DATA_IMAGE/Train/TREE/181.jpg
inflating: DATA_IMAGE/Train/TREE/182.jpeg
inflating: DATA_IMAGE/Train/TREE/183.jpeg
inflating: DATA_IMAGE/Train/TREE/184.jpeg
inflating: DATA_IMAGE/Train/TREE/185.jpeg
inflating: DATA_IMAGE/Train/TREE/186.jpg
inflating: DATA_IMAGE/Train/TREE/187.jpg
inflating: DATA_IMAGE/Train/TREE/188.jpeg
inflating: DATA_IMAGE/Train/TREE/189.jpg
inflating: DATA_IMAGE/Train/TREE/190.jpeg
inflating: DATA_IMAGE/Train/TREE/191.jpg
inflating: DATA_IMAGE/Train/TREE/192.jpg
inflating: DATA_IMAGE/Train/TREE/193.jpg
inflating: DATA_IMAGE/Train/TREE/194.jpeg
inflating: DATA_IMAGE/Train/TREE/195.jpg
inflating: DATA_IMAGE/Train/TREE/196.jpg
inflating: DATA_IMAGE/Train/TREE/197.jpeg
inflating: DATA_IMAGE/Train/TREE/198.jpeg
inflating: DATA_IMAGE/Train/TREE/199.jpg
inflating: DATA_IMAGE/Train/TREE/2.jpg
inflating: DATA_IMAGE/Train/TREE/200.jpeg
inflating: DATA_IMAGE/Train/TREE/3.jpg
```

Exploring the Images

First we would like to see what the images look like. So we would store 4 images from every directory inside test and train

In [52]:
```
import os
test_path = './DATA_IMAGE/Test'
train_path = './DATA_IMAGE/Train'
test_dir = os.listdir(test_path)
train_dir = os.listdir(train_path)
labels = test_dir

image_show = []   # contains the directory to show images

# get 4 images from every directory inside test directory and train directoy
# and add to image_show list
def get_images(test_or_train_dir, test_or_train_path):
  for index, dir in enumerate(test_or_train_dir):
    image_show_dir = f"{test_or_train_path}/{dir}"
    image_inside_dir = os.listdir(image_show_dir)

    for i in range(0,4):
      image_show.append(image_show_dir + '/'+image_inside_dir[i])

get_images(test_dir, test_path)
get_images(train_dir, train_path)

image_show
```

Out[52]:
```
['./DATA_IMAGE/Test/TREE/38.jpg',
 './DATA_IMAGE/Test/TREE/77.jpeg',
 './DATA_IMAGE/Test/TREE/91.jpg',
 './DATA_IMAGE/Test/TREE/58.jpg',
 './DATA_IMAGE/Test/CAR/38.jpg',
 './DATA_IMAGE/Test/CAR/91.jpg',
 './DATA_IMAGE/Test/CAR/3.jpg',
 './DATA_IMAGE/Test/CAR/58.jpg',
 './DATA_IMAGE/Train/TREE/151.jpg',
 './DATA_IMAGE/Train/TREE/180.jpg',
 './DATA_IMAGE/Train/TREE/166.jpg',
 './DATA_IMAGE/Train/TREE/3.jpg',
 './DATA_IMAGE/Train/CARS/151.jpg',
 './DATA_IMAGE/Train/CARS/180.jpg',
 './DATA_IMAGE/Train/CARS/166.jpg',
 './DATA_IMAGE/Train/CARS/105.jpg']
```

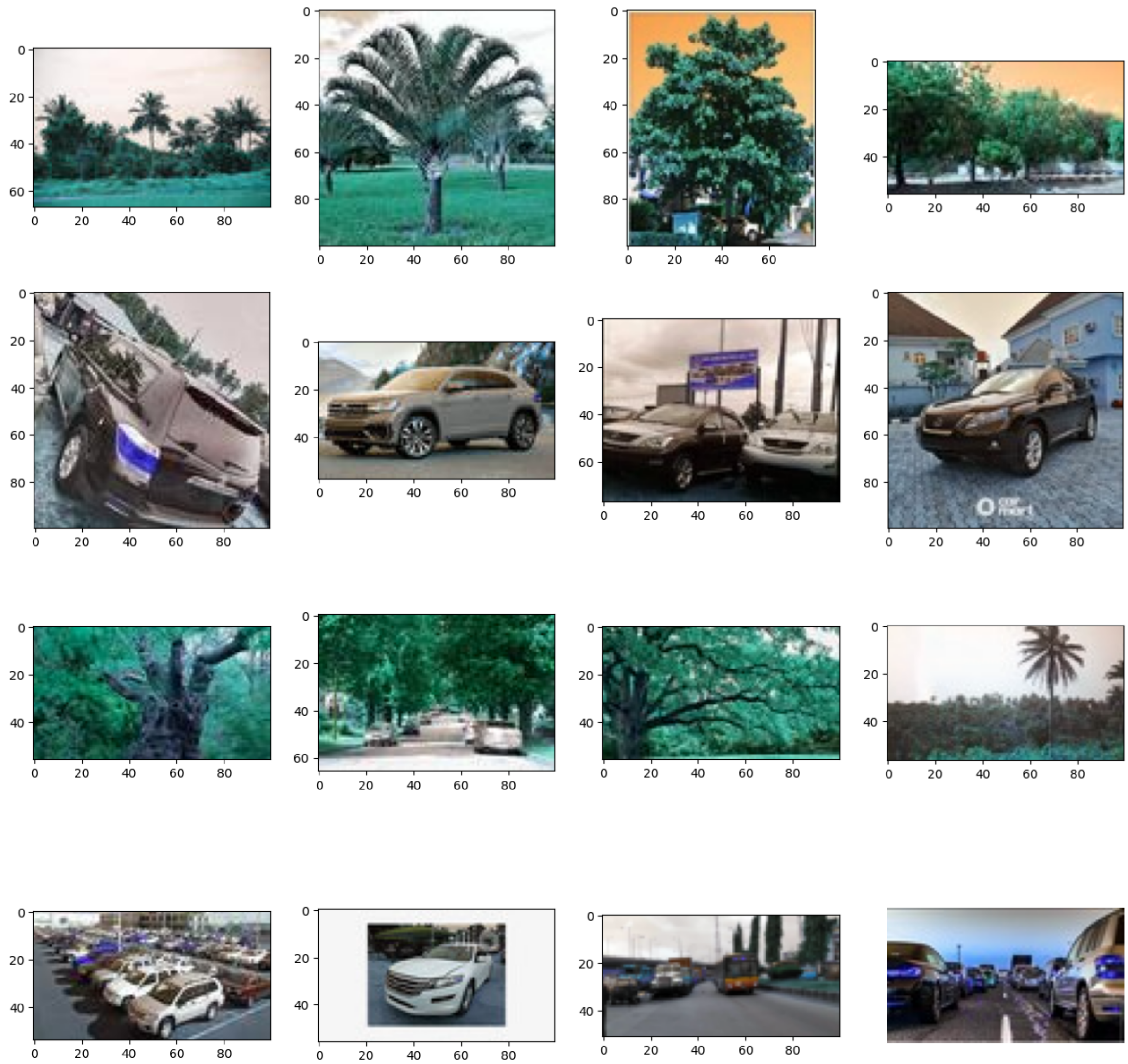
Displaying the Images

In [53]:

```
import matplotlib.pyplot as plt
import cv2 as cv

figure, axes = plt.subplots(4,4,figsize=(16,16))

for row in range (4):
  for col in range (4):
    index = row + 4 * col
    image_path = image_show[index]
    image = cv.imread(image_path)
    axes[col,row].imshow(image)
    plt.axis('off')
```



Here in this plot, the first row contains 4 sample of car images from test data. Second row shows 4 sample of tree images from test data. Thir row shows 4 sample of car images from train data. Fourth row shows 4 sample of tree images from train data. We can also see that the images displayed on this plot is of different sizes, so we have to resize the image before we can train and test the model.

Loading Train and Test Data

In [54]:
```
IMG_SIZE = (50,50)
BATCH_SIZE = 20
```

In [55]:
```
import tensorflow as tf

train_data = tf.keras.utils.image_dataset_from_directory(
    train_path,
    labels='inferred',
    label_mode = 'categorical',
    color_mode = 'grayscale',
    batch_size = BATCH_SIZE,
    image_size = IMG_SIZE
)

test_data = tf.keras.utils.image_dataset_from_directory(
    test_path,
    labels='inferred',
    label_mode = 'categorical',
    color_mode = 'grayscale',
    batch_size = BATCH_SIZE,
    image_size = IMG_SIZE
)

train_data_rgb = tf.keras.utils.image_dataset_from_directory(
    train_path,
    labels='inferred',
    label_mode = 'categorical',
    color_mode = 'rgb',
    batch_size = BATCH_SIZE,
    image_size = IMG_SIZE
)

test_data_rgb = tf.keras.utils.image_dataset_from_directory(
    test_path,
    labels='inferred',
    label_mode = 'categorical',
    color_mode = 'rgb',
    batch_size = BATCH_SIZE,
    image_size = IMG_SIZE
)
```

```
Found 200 files belonging to 2 classes.
Found 200 files belonging to 2 classes.
Found 200 files belonging to 2 classes.
Found 200 files belonging to 2 classes.
```

Here we have 4 types of dataset. train_data and test_data contains the images that are stored as grayscale. This data is used for sequential and CNN models. Then we have train_data_rgb and test_data_rgb which are used for Pretrained Model and Transfer Learning in Google's MobileNetV2 model.

Sequential Model

In this secton we will be creating a sequential model using our train_data that we loaded in the previous section and test it using test_data

First we create the model:

In [56]:
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=IMG_SIZE),
  tf.keras.layers.Dense(350, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(350, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(2, activation='softmax'),
])
```

Then we compile the model with loss, optimizer and metrics argument

In [57]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

Then we fit the model with training data and perform tests on it

In [58]:
```
history = model.fit(x=train_data,
                    epochs=10,
                    verbose=1,
                    validation_data=test_data)
```

```
Epoch 1/10
10/10 [==============================] - 3s 89ms/step - loss: 824.9829 - accuracy:
0.4750 - val_loss: 377.9778 - val_accuracy: 0.5000
Epoch 2/10
10/10 [==============================] - 1s 49ms/step - loss: 169.3825 - accuracy:
0.4700 - val_loss: 119.7350 - val_accuracy: 0.5000
Epoch 3/10
10/10 [==============================] - 1s 43ms/step - loss: 80.1783 - accuracy:
0.5050 - val_loss: 21.9016 - val_accuracy: 0.5000
Epoch 4/10
10/10 [==============================] - 1s 59ms/step - loss: 51.8897 - accuracy:
0.5050 - val_loss: 51.1312 - val_accuracy: 0.5000
Epoch 5/10
10/10 [==============================] - 0s 38ms/step - loss: 42.1113 - accuracy:
0.5350 - val_loss: 31.0430 - val_accuracy: 0.5000
Epoch 6/10
10/10 [==============================] - 1s 47ms/step - loss: 20.9212 - accuracy:
0.5150 - val_loss: 12.2933 - val_accuracy: 0.5050
Epoch 7/10
10/10 [==============================] - 1s 72ms/step - loss: 7.5751 - accuracy: 0.
5350 - val_loss: 0.7088 - val_accuracy: 0.6650
Epoch 8/10
10/10 [==============================] - 1s 63ms/step - loss: 0.9719 - accuracy: 0.
5750 - val_loss: 0.8306 - val_accuracy: 0.5200
Epoch 9/10
10/10 [==============================] - 1s 72ms/step - loss: 0.9815 - accuracy: 0.
5450 - val_loss: 0.7733 - val_accuracy: 0.5700
Epoch 10/10
10/10 [==============================] - 0s 35ms/step - loss: 0.7357 - accuracy: 0.
5400 - val_loss: 0.7785 - val_accuracy: 0.5300
```

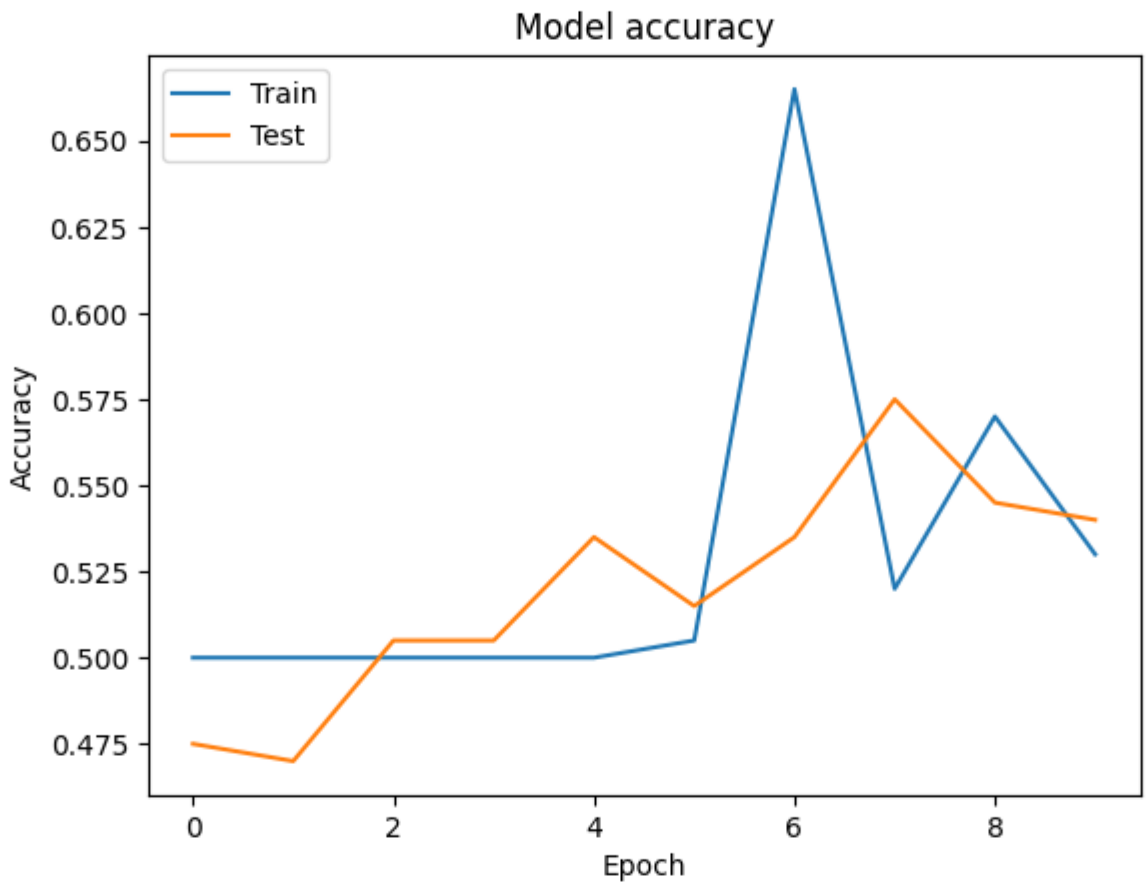
In [59]:
```
history.history.keys()
```

Out[59]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

We can display the Accuracy of training and testing data as a graph

In [60]:
```
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



As we can see that the highest training accuract was 57.5% and highest testing accuracy was 65%

CNN

In this secton we will be creating a sequential model using our train_data that we loaded in the previous section and test it using test_data

First we create the model:

In [61]:
```
model = tf.keras.models.Sequential(
    [
        tf.keras.Input(shape=(50, 50, 1)),
        tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(2, activation="softmax"),
    ]
)
```

let take a look at the summary of the model

In [62]:
```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 48, 48, 32)        320

 max_pooling2d_2 (MaxPooling  (None, 24, 24, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 22, 22, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 11, 11, 64)       0
 2D)

 flatten_4 (Flatten)         (None, 7744)              0

 dropout_9 (Dropout)         (None, 7744)              0

 dense_12 (Dense)            (None, 2)                 15490

=================================================================
Total params: 34,306
Trainable params: 34,306
Non-trainable params: 0
_________________________________________________________________
```

we compile the model with loss, optimizer and metrics argument and then we we fit the model with training data and perform tests on it.

In [63]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

history = model.fit(train_data,
                    epochs=30,
                    verbose=1,
                    validation_data=test_data)
```

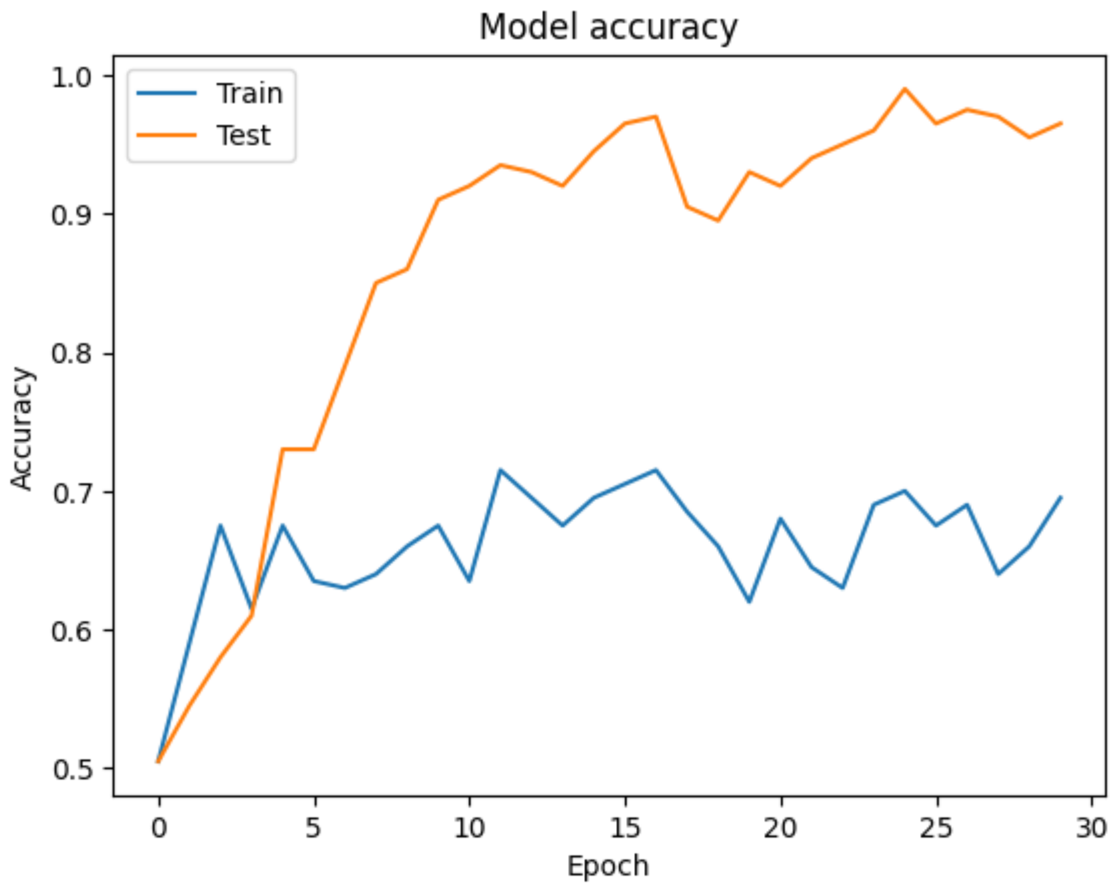
```
Epoch 1/30
10/10 [==============================] - 3s 216ms/step - loss: 16.8179 - accuracy:
0.5050 - val_loss: 6.8577 - val_accuracy: 0.5050
Epoch 2/30
10/10 [==============================] - 2s 193ms/step - loss: 3.1828 - accuracy:
0.5450 - val_loss: 0.9770 - val_accuracy: 0.5900
Epoch 3/30
10/10 [==============================] - 1s 93ms/step - loss: 1.1117 - accuracy: 0.
5800 - val_loss: 0.6486 - val_accuracy: 0.6750
Epoch 4/30
10/10 [==============================] - 1s 105ms/step - loss: 0.8684 - accuracy:
0.6100 - val_loss: 0.7321 - val_accuracy: 0.6150
Epoch 5/30
10/10 [==============================] - 1s 102ms/step - loss: 0.5582 - accuracy:
0.7300 - val_loss: 0.6196 - val_accuracy: 0.6750
Epoch 6/30
10/10 [==============================] - 1s 95ms/step - loss: 0.4793 - accuracy: 0.
7300 - val_loss: 0.6597 - val_accuracy: 0.6350
Epoch 7/30
10/10 [==============================] - 1s 92ms/step - loss: 0.4009 - accuracy: 0.
7900 - val_loss: 0.6515 - val_accuracy: 0.6300
Epoch 8/30
10/10 [==============================] - 1s 99ms/step - loss: 0.3272 - accuracy: 0.
8500 - val_loss: 0.6943 - val_accuracy: 0.6400
Epoch 9/30
10/10 [==============================] - 1s 104ms/step - loss: 0.3017 - accuracy:
0.8600 - val_loss: 0.6469 - val_accuracy: 0.6600
Epoch 10/30
10/10 [==============================] - 1s 95ms/step - loss: 0.2214 - accuracy: 0.
9100 - val_loss: 0.6371 - val_accuracy: 0.6750
Epoch 11/30
10/10 [==============================] - 1s 131ms/step - loss: 0.1771 - accuracy:
0.9200 - val_loss: 0.7335 - val_accuracy: 0.6350
Epoch 12/30
10/10 [==============================] - 1s 127ms/step - loss: 0.1691 - accuracy:
0.9350 - val_loss: 0.7231 - val_accuracy: 0.7150
Epoch 13/30
10/10 [==============================] - 1s 103ms/step - loss: 0.1966 - accuracy:
0.9300 - val_loss: 0.7140 - val_accuracy: 0.6950
Epoch 14/30
10/10 [==============================] - 1s 103ms/step - loss: 0.1791 - accuracy:
0.9200 - val_loss: 0.6793 - val_accuracy: 0.6750
Epoch 15/30
10/10 [==============================] - 1s 106ms/step - loss: 0.1611 - accuracy:
0.9450 - val_loss: 0.6601 - val_accuracy: 0.6950
Epoch 16/30
10/10 [==============================] - 1s 106ms/step - loss: 0.0962 - accuracy:
0.9650 - val_loss: 0.8474 - val_accuracy: 0.7050
Epoch 17/30
10/10 [==============================] - 1s 108ms/step - loss: 0.1037 - accuracy:
0.9700 - val_loss: 0.7900 - val_accuracy: 0.7150
Epoch 18/30
10/10 [==============================] - 1s 92ms/step - loss: 0.2210 - accuracy: 0.
9050 - val_loss: 0.7424 - val_accuracy: 0.6850
Epoch 19/30
10/10 [==============================] - 2s 162ms/step - loss: 0.2395 - accuracy:
0.8950 - val_loss: 0.6835 - val_accuracy: 0.6600
Epoch 20/30
10/10 [==============================] - 3s 291ms/step - loss: 0.1883 - accuracy:
```

```
0.9300 - val_loss: 0.7333 - val_accuracy: 0.6200
Epoch 21/30
10/10 [==============================] - 3s 259ms/step - loss: 0.2050 - accuracy: 
0.9200 - val_loss: 0.7879 - val_accuracy: 0.6800
Epoch 22/30
10/10 [==============================] - 1s 96ms/step - loss: 0.1347 - accuracy: 0.
9400 - val_loss: 0.9221 - val_accuracy: 0.6450
Epoch 23/30
10/10 [==============================] - 1s 90ms/step - loss: 0.1168 - accuracy: 0.
9500 - val_loss: 0.9195 - val_accuracy: 0.6300
Epoch 24/30
10/10 [==============================] - 1s 98ms/step - loss: 0.1293 - accuracy: 0.
9600 - val_loss: 0.8087 - val_accuracy: 0.6900
Epoch 25/30
10/10 [==============================] - 1s 108ms/step - loss: 0.0676 - accuracy: 
0.9900 - val_loss: 0.8511 - val_accuracy: 0.7000
Epoch 26/30
10/10 [==============================] - 1s 105ms/step - loss: 0.0847 - accuracy: 
0.9650 - val_loss: 0.8205 - val_accuracy: 0.6750
Epoch 27/30
10/10 [==============================] - 1s 94ms/step - loss: 0.0988 - accuracy: 0.
9750 - val_loss: 0.7993 - val_accuracy: 0.6900
Epoch 28/30
10/10 [==============================] - 2s 181ms/step - loss: 0.0790 - accuracy: 
0.9700 - val_loss: 0.9963 - val_accuracy: 0.6400
Epoch 29/30
10/10 [==============================] - 1s 131ms/step - loss: 0.1129 - accuracy: 
0.9550 - val_loss: 1.1760 - val_accuracy: 0.6600
Epoch 30/30
10/10 [==============================] - 1s 102ms/step - loss: 0.0741 - accuracy: 
0.9650 - val_loss: 1.2625 - val_accuracy: 0.6950
```

We can display the Accuracy of training and testing data as a graph

In [64]:

```
# Plot training & validation accuracy values
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



As we can see that the highest training accuracy was 99% and highest testing accuracy was 71.50%. This is a significant increase from sequential model.

Pretrained Model and Transfer Learning

Note: Few of the code snippets used in this section is driectly taken from official website: https://www.tensorflow.org/tutorials/images/transfer_learning

In this section we will perform transfer learning on pretrained model. The pretrained model would be google's MobileNetV2

First we will like to use data augmentation because our dataset is small.

In [65]:
```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip('horizontal'),
  tf.keras.layers.RandomRotation(0.2),
])
```

Let see what data augmentation can do the images

```
In [66]: for image, _ in train_data_rgb.take(1):
           plt.figure(figsize=(10, 10))
           first_image = image[0]

           for i in range(9):
             ax = plt.subplot(3, 3, i + 1)
             augmented_image = data_augmentation(tf.expand_dims(first_image, 0))
             plt.imshow(augmented_image[0] / 255)
             plt.axis('off')
```

Add classification head

Since we are going to use MobileNetV2 as the base model, we need the pixel values as [-1,1]. Currently the pixel values stored are between [0,255]. To rescale the pixel values we will use preprocess_input

```
In [67]: preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
```

Now we will create base model based on MobileNetV2 from google with weights=imagenet

In [68]:
```
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [9
6, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the de
fault.
```

In [69]:
```
image_batch, label_batch = next(iter(train_data_rgb))
feature_batch = base_model(image_batch)
print(feature_batch.shape)
```

```
(20, 2, 2, 1280)
```

We will be freezing the given layer so that it prevent the weight from updating during the training. This can be done by setting the `layer.trainable = False` . Then we can see the summay of the model

In [70]:
```
base_model.trainable = False
base_model.summary()
```

```
Model: "mobilenetv2_1.00_224"
__________________________________________________________________________________
________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
================
 input_8 (InputLayer)           [(None, 50, 50, 3)]  0           []

 Conv1 (Conv2D)                 (None, 25, 25, 32)   864         ['input_8[0][0]']

 bn_Conv1 (BatchNormalization)  (None, 25, 25, 32)   128         ['Conv1[0][0]']

 Conv1_relu (ReLU)              (None, 25, 25, 32)   0           ['bn_Conv1[0][0]']

 expanded_conv_depthwise (Depth  (None, 25, 25, 32)  288         ['Conv1_relu
[0][0]']
 wiseConv2D)

 expanded_conv_depthwise_BN (Ba  (None, 25, 25, 32)  128         ['expanded_conv_de
pthwise[0][0]']
 tchNormalization)

 expanded_conv_depthwise_relu (  (None, 25, 25, 32)  0           ['expanded_conv_de
pthwise_BN[0][0
 ReLU)                                                            ]']

 expanded_conv_project (Conv2D)  (None, 25, 25, 16)  512         ['expanded_conv_de
pthwise_relu[0]
                                                                  [0]']

 expanded_conv_project_BN (Batc  (None, 25, 25, 16)  64          ['expanded_conv_pr
oject[0][0]']
 hNormalization)

 block_1_expand (Conv2D)        (None, 25, 25, 96)   1536        ['expanded_conv_pr
oject_BN[0][0]'
                                                                  ]

 block_1_expand_BN (BatchNormal  (None, 25, 25, 96)  384         ['block_1_expand
[0][0]']
 ization)

 block_1_expand_relu (ReLU)     (None, 25, 25, 96)   0           ['block_1_expand_B
N[0][0]']

 block_1_pad (ZeroPadding2D)    (None, 27, 27, 96)   0           ['block_1_expand_r
elu[0][0]']

 block_1_depthwise (DepthwiseCo  (None, 13, 13, 96)  864         ['block_1_pad
[0][0]']
 nv2D)

 block_1_depthwise_BN (BatchNor  (None, 13, 13, 96)  384         ['block_1_depthwis
e[0][0]']
 malization)

 block_1_depthwise_relu (ReLU)  (None, 13, 13, 96)   0           ['block_1_depthwis
e_BN[0][0]']
```

```
 block_1_project (Conv2D)       (None, 13, 13, 24)   2304        ['block_1_depthwis
e_relu[0][0]']

 block_1_project_BN (BatchNorma  (None, 13, 13, 24)  96          ['block_1_project
[0][0]']
 lization)

 block_2_expand (Conv2D)        (None, 13, 13, 144)  3456        ['block_1_project_
BN[0][0]']

 block_2_expand_BN (BatchNormal  (None, 13, 13, 144)  576        ['block_2_expand
[0][0]']
 ization)

 block_2_expand_relu (ReLU)     (None, 13, 13, 144)  0           ['block_2_expand_B
N[0][0]']

 block_2_depthwise (DepthwiseCo  (None, 13, 13, 144)  1296       ['block_2_expand_r
elu[0][0]']
 nv2D)

 block_2_depthwise_BN (BatchNor  (None, 13, 13, 144)  576        ['block_2_depthwis
e[0][0]']
 malization)

 block_2_depthwise_relu (ReLU)  (None, 13, 13, 144)  0           ['block_2_depthwis
e_BN[0][0]']

 block_2_project (Conv2D)       (None, 13, 13, 24)   3456        ['block_2_depthwis
e_relu[0][0]']

 block_2_project_BN (BatchNorma  (None, 13, 13, 24)  96          ['block_2_project
[0][0]']
 lization)

 block_2_add (Add)              (None, 13, 13, 24)   0           ['block_1_project_
BN[0][0]',
                                                                  'block_2_project_
BN[0][0]']

 block_3_expand (Conv2D)        (None, 13, 13, 144)  3456        ['block_2_add
[0][0]']

 block_3_expand_BN (BatchNormal  (None, 13, 13, 144)  576        ['block_3_expand
[0][0]']
 ization)

 block_3_expand_relu (ReLU)     (None, 13, 13, 144)  0           ['block_3_expand_B
N[0][0]']

 block_3_pad (ZeroPadding2D)    (None, 15, 15, 144)  0           ['block_3_expand_r
elu[0][0]']

 block_3_depthwise (DepthwiseCo  (None, 7, 7, 144)   1296        ['block_3_pad
[0][0]']
 nv2D)

 block_3_depthwise_BN (BatchNor  (None, 7, 7, 144)   576         ['block_3_depthwis
e[0][0]']
```

```
 malization)

 block_3_depthwise_relu (ReLU)  (None, 7, 7, 144)    0           ['block_3_depthwis
e_BN[0][0]']

 block_3_project (Conv2D)       (None, 7, 7, 32)     4608        ['block_3_depthwis
e_relu[0][0]']

 block_3_project_BN (BatchNorma  (None, 7, 7, 32)    128         ['block_3_project
[0][0]']
 lization)

 block_4_expand (Conv2D)        (None, 7, 7, 192)    6144        ['block_3_project_
BN[0][0]']

 block_4_expand_BN (BatchNormal  (None, 7, 7, 192)   768         ['block_4_expand
[0][0]']
 ization)

 block_4_expand_relu (ReLU)     (None, 7, 7, 192)    0           ['block_4_expand_B
N[0][0]']

 block_4_depthwise (DepthwiseCo  (None, 7, 7, 192)   1728        ['block_4_expand_r
elu[0][0]']
 nv2D)

 block_4_depthwise_BN (BatchNor  (None, 7, 7, 192)   768         ['block_4_depthwis
e[0][0]']
 malization)

 block_4_depthwise_relu (ReLU)  (None, 7, 7, 192)    0           ['block_4_depthwis
e_BN[0][0]']

 block_4_project (Conv2D)       (None, 7, 7, 32)     6144        ['block_4_depthwis
e_relu[0][0]']

 block_4_project_BN (BatchNorma  (None, 7, 7, 32)    128         ['block_4_project
[0][0]']
 lization)

 block_4_add (Add)              (None, 7, 7, 32)     0           ['block_3_project_
BN[0][0]',

                                                                  'block_4_project_

BN[0][0]']

 block_5_expand (Conv2D)        (None, 7, 7, 192)    6144        ['block_4_add
[0][0]']

 block_5_expand_BN (BatchNormal  (None, 7, 7, 192)   768         ['block_5_expand
[0][0]']
 ization)

 block_5_expand_relu (ReLU)     (None, 7, 7, 192)    0           ['block_5_expand_B
N[0][0]']

 block_5_depthwise (DepthwiseCo  (None, 7, 7, 192)   1728        ['block_5_expand_r
elu[0][0]']
 nv2D)
```

```
 block_5_depthwise_BN (BatchNor  (None, 7, 7, 192)   768         ['block_5_depthwis
e[0][0]']
 malization)

 block_5_depthwise_relu (ReLU)  (None, 7, 7, 192)   0           ['block_5_depthwis
e_BN[0][0]']

 block_5_project (Conv2D)       (None, 7, 7, 32)    6144        ['block_5_depthwis
e_relu[0][0]']

 block_5_project_BN (BatchNorma  (None, 7, 7, 32)   128         ['block_5_project
[0][0]']
 lization)

 block_5_add (Add)              (None, 7, 7, 32)    0           ['block_4_add
[0][0]',
                                                                 'block_5_project_
BN[0][0]']

 block_6_expand (Conv2D)        (None, 7, 7, 192)   6144        ['block_5_add
[0][0]']

 block_6_expand_BN (BatchNormal  (None, 7, 7, 192)  768         ['block_6_expand
[0][0]']
 ization)

 block_6_expand_relu (ReLU)     (None, 7, 7, 192)   0           ['block_6_expand_B
N[0][0]']

 block_6_pad (ZeroPadding2D)    (None, 9, 9, 192)   0           ['block_6_expand_r
elu[0][0]']

 block_6_depthwise (DepthwiseCo  (None, 4, 4, 192)  1728        ['block_6_pad
[0][0]']
 nv2D)

 block_6_depthwise_BN (BatchNor  (None, 4, 4, 192)  768         ['block_6_depthwis
e[0][0]']
 malization)

 block_6_depthwise_relu (ReLU)  (None, 4, 4, 192)   0           ['block_6_depthwis
e_BN[0][0]']

 block_6_project (Conv2D)       (None, 4, 4, 64)    12288       ['block_6_depthwis
e_relu[0][0]']

 block_6_project_BN (BatchNorma  (None, 4, 4, 64)   256         ['block_6_project
[0][0]']
 lization)

 block_7_expand (Conv2D)        (None, 4, 4, 384)   24576       ['block_6_project_
BN[0][0]']

 block_7_expand_BN (BatchNormal  (None, 4, 4, 384)  1536        ['block_7_expand
[0][0]']
 ization)

 block_7_expand_relu (ReLU)     (None, 4, 4, 384)   0           ['block_7_expand_B
N[0][0]']
```

```
 block_7_depthwise (DepthwiseCo  (None, 4, 4, 384)   3456        ['block_7_expand_r
elu[0][0]']
 nv2D)

 block_7_depthwise_BN (BatchNor  (None, 4, 4, 384)   1536        ['block_7_depthwis
e[0][0]']
 malization)

 block_7_depthwise_relu (ReLU)  (None, 4, 4, 384)    0           ['block_7_depthwis
e_BN[0][0]']

 block_7_project (Conv2D)       (None, 4, 4, 64)     24576       ['block_7_depthwis
e_relu[0][0]']

 block_7_project_BN (BatchNorma  (None, 4, 4, 64)    256         ['block_7_project
[0][0]']
 lization)

 block_7_add (Add)              (None, 4, 4, 64)     0           ['block_6_project_
BN[0][0]',
                                                                  'block_7_project_
BN[0][0]']

 block_8_expand (Conv2D)        (None, 4, 4, 384)    24576       ['block_7_add
[0][0]']

 block_8_expand_BN (BatchNormal  (None, 4, 4, 384)   1536        ['block_8_expand
[0][0]']
 ization)

 block_8_expand_relu (ReLU)     (None, 4, 4, 384)    0           ['block_8_expand_B
N[0][0]']

 block_8_depthwise (DepthwiseCo  (None, 4, 4, 384)   3456        ['block_8_expand_r
elu[0][0]']
 nv2D)

 block_8_depthwise_BN (BatchNor  (None, 4, 4, 384)   1536        ['block_8_depthwis
e[0][0]']
 malization)

 block_8_depthwise_relu (ReLU)  (None, 4, 4, 384)    0           ['block_8_depthwis
e_BN[0][0]']

 block_8_project (Conv2D)       (None, 4, 4, 64)     24576       ['block_8_depthwis
e_relu[0][0]']

 block_8_project_BN (BatchNorma  (None, 4, 4, 64)    256         ['block_8_project
[0][0]']
 lization)

 block_8_add (Add)              (None, 4, 4, 64)     0           ['block_7_add
[0][0]',
                                                                  'block_8_project_
BN[0][0]']

 block_9_expand (Conv2D)        (None, 4, 4, 384)    24576       ['block_9_add
[0][0]']
```

```
 block_9_expand_BN (BatchNormal  (None, 4, 4, 384)   1536        ['block_9_expand
[0][0]']
 ization)

 block_9_expand_relu (ReLU)     (None, 4, 4, 384)   0           ['block_9_expand_B
N[0][0]']

 block_9_depthwise (DepthwiseCo  (None, 4, 4, 384)   3456        ['block_9_expand_r
elu[0][0]']
 nv2D)

 block_9_depthwise_BN (BatchNor  (None, 4, 4, 384)   1536        ['block_9_depthwis
e[0][0]']
 malization)

 block_9_depthwise_relu (ReLU)  (None, 4, 4, 384)   0           ['block_9_depthwis
e_BN[0][0]']

 block_9_project (Conv2D)       (None, 4, 4, 64)    24576       ['block_9_depthwis
e_relu[0][0]']

 block_9_project_BN (BatchNorma  (None, 4, 4, 64)    256         ['block_9_project
[0][0]']
 lization)

 block_9_add (Add)              (None, 4, 4, 64)    0           ['block_8_add
[0][0]',
                                                                  'block_9_project_
BN[0][0]']

 block_10_expand (Conv2D)       (None, 4, 4, 384)   24576       ['block_9_add
[0][0]']

 block_10_expand_BN (BatchNorma  (None, 4, 4, 384)   1536        ['block_10_expand
[0][0]']
 lization)

 block_10_expand_relu (ReLU)    (None, 4, 4, 384)   0           ['block_10_expand_
BN[0][0]']

 block_10_depthwise (DepthwiseC  (None, 4, 4, 384)   3456        ['block_10_expand_
relu[0][0]']
 onv2D)

 block_10_depthwise_BN (BatchNo  (None, 4, 4, 384)   1536        ['block_10_depthwi
se[0][0]']
 rmalization)

 block_10_depthwise_relu (ReLU)  (None, 4, 4, 384)   0           ['block_10_depthwi
se_BN[0][0]']

 block_10_project (Conv2D)      (None, 4, 4, 96)    36864       ['block_10_depthwi
se_relu[0][0]']

 block_10_project_BN (BatchNorm  (None, 4, 4, 96)    384         ['block_10_project
[0][0]']
 alization)
```

```
 block_11_expand (Conv2D)       (None, 4, 4, 576)    55296       ['block_10_project
_BN[0][0]']

 block_11_expand_BN (BatchNorma  (None, 4, 4, 576)   2304        ['block_11_expand
[0][0]']
 lization)

 block_11_expand_relu (ReLU)    (None, 4, 4, 576)    0           ['block_11_expand_
BN[0][0]']

 block_11_depthwise (DepthwiseC  (None, 4, 4, 576)   5184        ['block_11_expand_
relu[0][0]']
 onv2D)

 block_11_depthwise_BN (BatchNo  (None, 4, 4, 576)   2304        ['block_11_depthwi
se[0][0]']
 rmalization)

 block_11_depthwise_relu (ReLU)  (None, 4, 4, 576)   0           ['block_11_depthwi
se_BN[0][0]']

 block_11_project (Conv2D)      (None, 4, 4, 96)     55296       ['block_11_depthwi
se_relu[0][0]']

 block_11_project_BN (BatchNorm  (None, 4, 4, 96)    384         ['block_11_project
[0][0]']
 alization)

 block_11_add (Add)             (None, 4, 4, 96)     0           ['block_10_project
_BN[0][0]',
                                                                  'block_11_project
_BN[0][0]']

 block_12_expand (Conv2D)       (None, 4, 4, 576)    55296       ['block_11_add
[0][0]']

 block_12_expand_BN (BatchNorma  (None, 4, 4, 576)   2304        ['block_12_expand
[0][0]']
 lization)

 block_12_expand_relu (ReLU)    (None, 4, 4, 576)    0           ['block_12_expand_
BN[0][0]']

 block_12_depthwise (DepthwiseC  (None, 4, 4, 576)   5184        ['block_12_expand_
relu[0][0]']
 onv2D)

 block_12_depthwise_BN (BatchNo  (None, 4, 4, 576)   2304        ['block_12_depthwi
se[0][0]']
 rmalization)

 block_12_depthwise_relu (ReLU)  (None, 4, 4, 576)   0           ['block_12_depthwi
se_BN[0][0]']

 block_12_project (Conv2D)      (None, 4, 4, 96)     55296       ['block_12_depthwi
se_relu[0][0]']

 block_12_project_BN (BatchNorm  (None, 4, 4, 96)    384         ['block_12_project
[0][0]']
```

```
 alization)

 block_12_add (Add)             (None, 4, 4, 96)     0            ['block_11_add
[0][0]',
                                                                   'block_12_project
_BN[0][0]']

 block_13_expand (Conv2D)       (None, 4, 4, 576)    55296        ['block_12_add
[0][0]']

 block_13_expand_BN (BatchNorma  (None, 4, 4, 576)   2304         ['block_13_expand
[0][0]']
 lization)

 block_13_expand_relu (ReLU)    (None, 4, 4, 576)    0            ['block_13_expand_
BN[0][0]']

 block_13_pad (ZeroPadding2D)   (None, 5, 5, 576)    0            ['block_13_expand_
relu[0][0]']

 block_13_depthwise (DepthwiseC  (None, 2, 2, 576)   5184         ['block_13_pad
[0][0]']
 onv2D)

 block_13_depthwise_BN (BatchNo  (None, 2, 2, 576)   2304         ['block_13_depthwi
se[0][0]']
 rmalization)

 block_13_depthwise_relu (ReLU)  (None, 2, 2, 576)   0            ['block_13_depthwi
se_BN[0][0]']

 block_13_project (Conv2D)      (None, 2, 2, 160)    92160        ['block_13_depthwi
se_relu[0][0]']

 block_13_project_BN (BatchNorm  (None, 2, 2, 160)   640          ['block_13_project
[0][0]']
 alization)

 block_14_expand (Conv2D)       (None, 2, 2, 960)    153600       ['block_13_project
_BN[0][0]']

 block_14_expand_BN (BatchNorma  (None, 2, 2, 960)   3840         ['block_14_expand
[0][0]']
 lization)

 block_14_expand_relu (ReLU)    (None, 2, 2, 960)    0            ['block_14_expand_
BN[0][0]']

 block_14_depthwise (DepthwiseC  (None, 2, 2, 960)   8640         ['block_14_expand_
relu[0][0]']
 onv2D)

 block_14_depthwise_BN (BatchNo  (None, 2, 2, 960)   3840         ['block_14_depthwi
se[0][0]']
 rmalization)

 block_14_depthwise_relu (ReLU)  (None, 2, 2, 960)   0            ['block_14_depthwi
se_BN[0][0]']
```

```
 block_14_project (Conv2D)      (None, 2, 2, 160)    153600      ['block_14_depthwi
se_relu[0][0]']

 block_14_project_BN (BatchNorm  (None, 2, 2, 160)   640         ['block_14_project
[0][0]']
 alization)

 block_14_add (Add)             (None, 2, 2, 160)    0           ['block_13_project
_BN[0][0]',
                                                                  'block_14_project
_BN[0][0]']

 block_15_expand (Conv2D)       (None, 2, 2, 960)    153600      ['block_14_add
[0][0]']

 block_15_expand_BN (BatchNorma  (None, 2, 2, 960)   3840        ['block_15_expand
[0][0]']
 lization)

 block_15_expand_relu (ReLU)    (None, 2, 2, 960)    0           ['block_15_expand_
BN[0][0]']

 block_15_depthwise (DepthwiseC  (None, 2, 2, 960)   8640        ['block_15_expand_
relu[0][0]']
 onv2D)

 block_15_depthwise_BN (BatchNo  (None, 2, 2, 960)   3840        ['block_15_depthwi
se[0][0]']
 rmalization)

 block_15_depthwise_relu (ReLU)  (None, 2, 2, 960)   0           ['block_15_depthwi
se_BN[0][0]']

 block_15_project (Conv2D)      (None, 2, 2, 160)    153600      ['block_15_depthwi
se_relu[0][0]']

 block_15_project_BN (BatchNorm  (None, 2, 2, 160)   640         ['block_15_project
[0][0]']
 alization)

 block_15_add (Add)             (None, 2, 2, 160)    0           ['block_14_add
[0][0]',
                                                                  'block_15_project
_BN[0][0]']

 block_16_expand (Conv2D)       (None, 2, 2, 960)    153600      ['block_15_add
[0][0]']

 block_16_expand_BN (BatchNorma  (None, 2, 2, 960)   3840        ['block_16_expand
[0][0]']
 lization)

 block_16_expand_relu (ReLU)    (None, 2, 2, 960)    0           ['block_16_expand_
BN[0][0]']

 block_16_depthwise (DepthwiseC  (None, 2, 2, 960)   8640        ['block_16_expand_
relu[0][0]']
 onv2D)
```

```
 block_16_depthwise_BN (BatchNo  (None, 2, 2, 960)   3840        ['block_16_depthwi
se[0][0]']
 rmalization)

 block_16_depthwise_relu (ReLU)  (None, 2, 2, 960)   0           ['block_16_depthwi
se_BN[0][0]']

 block_16_project (Conv2D)      (None, 2, 2, 320)    307200      ['block_16_depthwi
se_relu[0][0]']

 block_16_project_BN (BatchNorm  (None, 2, 2, 320)   1280        ['block_16_project
[0][0]']
 alization)

 Conv_1 (Conv2D)                (None, 2, 2, 1280)   409600      ['block_16_project
_BN[0][0]']

 Conv_1_bn (BatchNormalization)  (None, 2, 2, 1280)  5120        ['Conv_1[0][0]']

 out_relu (ReLU)                (None, 2, 2, 1280)   0           ['Conv_1_bn
[0][0]']

==================================================================================
===============
Total params: 2,257,984
Trainable params: 0
Non-trainable params: 2,257,984
__________________________________________________________________________________
_______________
```

We will use `tf.keras.layers.GlobalAveragePooling2D` layer to convert the features to a single 1280-element vector per image.

In [71]:
```
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
feature_batch_average = global_average_layer(feature_batch)
print(feature_batch_average.shape)
```

```
(20, 1280)
```

we create a prediction layer and set it 2 as we need to predict for 2 different classes (cars and trees)

In [72]:
```
prediction_layer = tf.keras.layers.Dense(2) # this part was changed
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)
```

```
(20, 2)
```

Now we build the model by chaining data augmentation and rescalling and other feature extractor layers.

In [73]:
```python
inputs = tf.keras.Input(shape=(50, 50, 3))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

Now we compile the model

In [74]:
```python
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [75]:
```python
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_9 (InputLayer)        [(None, 50, 50, 3)]       0

 sequential_6 (Sequential)   (None, 50, 50, 3)         0

 tf.math.truediv_2 (TFOpLamb  (None, 50, 50, 3)        0
 da)

 tf.math.subtract_2 (TFOpLam  (None, 50, 50, 3)        0
 bda)

 mobilenetv2_1.00_224 (Funct  (None, 2, 2, 1280)       2257984
 ional)

 global_average_pooling2d_1   (None, 1280)             0
 (GlobalAveragePooling2D)

 dropout_10 (Dropout)        (None, 1280)              0

 dense_13 (Dense)            (None, 2)                 2562

=================================================================
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
_________________________________________________________________
```

As we can see that there are 2.5 million total parameters and among them 2,562 parameters are trainable in the dense layer. These are divided between two `tf.Variable` objects, the weights and biases

In [76]:
```python
len(model.trainable_variables)
```

Out[76]: 2

Now we can train the model with train_data_rgb and test it on test_data_rgb.

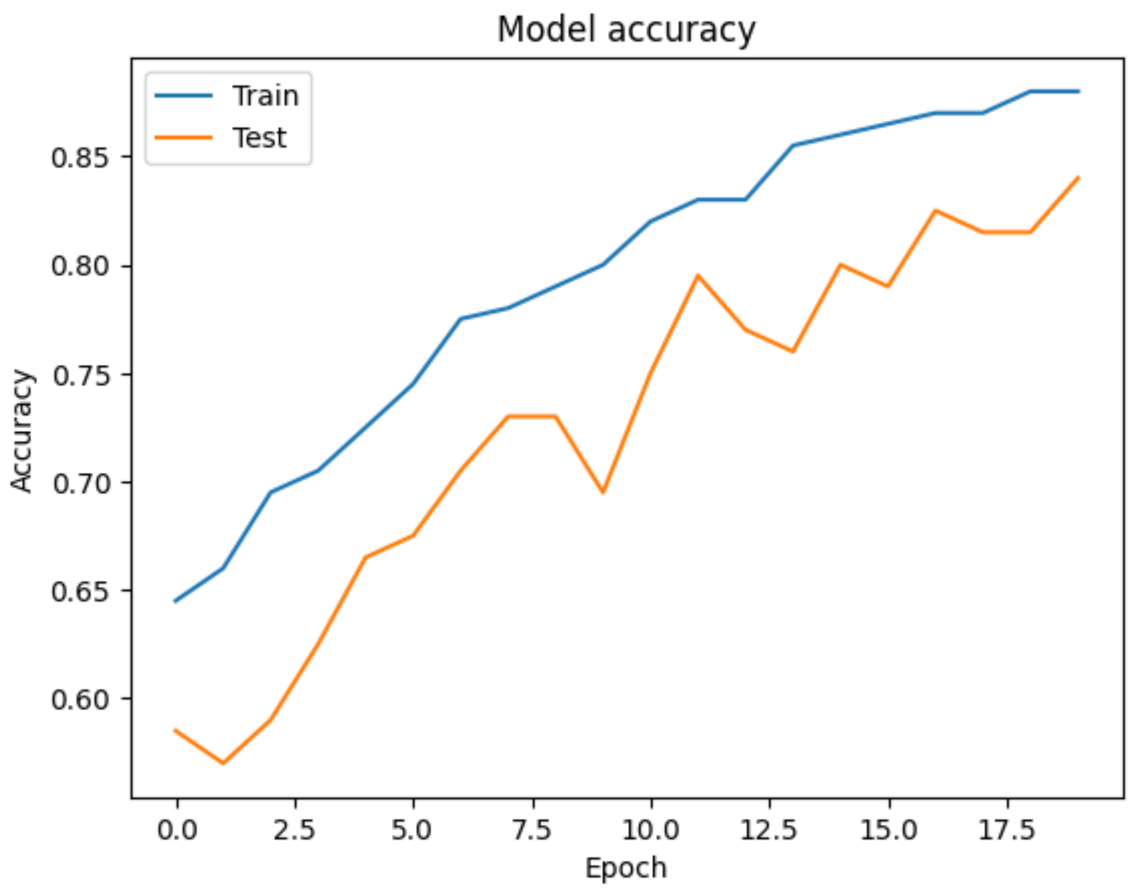
In [77]:
```
history = model.fit(train_data_rgb,
                    epochs=20,
                    validation_data=test_data_rgb)
```

```
Epoch 1/20
10/10 [==============================] - 12s 490ms/step - loss: 0.8589 - accuracy:
0.5850 - val_loss: 0.7904 - val_accuracy: 0.6450
Epoch 2/20
10/10 [==============================] - 2s 200ms/step - loss: 0.8555 - accuracy:
0.5700 - val_loss: 0.7487 - val_accuracy: 0.6600
Epoch 3/20
10/10 [==============================] - 2s 236ms/step - loss: 0.7462 - accuracy:
0.5900 - val_loss: 0.7119 - val_accuracy: 0.6950
Epoch 4/20
10/10 [==============================] - 2s 153ms/step - loss: 0.7445 - accuracy:
0.6250 - val_loss: 0.6836 - val_accuracy: 0.7050
Epoch 5/20
10/10 [==============================] - 1s 135ms/step - loss: 0.6883 - accuracy:
0.6650 - val_loss: 0.6557 - val_accuracy: 0.7250
Epoch 6/20
10/10 [==============================] - 1s 133ms/step - loss: 0.6823 - accuracy:
0.6750 - val_loss: 0.6329 - val_accuracy: 0.7450
Epoch 7/20
10/10 [==============================] - 1s 118ms/step - loss: 0.6628 - accuracy:
0.7050 - val_loss: 0.6129 - val_accuracy: 0.7750
Epoch 8/20
10/10 [==============================] - 1s 125ms/step - loss: 0.6281 - accuracy:
0.7300 - val_loss: 0.5971 - val_accuracy: 0.7800
Epoch 9/20
10/10 [==============================] - 1s 133ms/step - loss: 0.6469 - accuracy:
0.7300 - val_loss: 0.5816 - val_accuracy: 0.7900
Epoch 10/20
10/10 [==============================] - 2s 163ms/step - loss: 0.6316 - accuracy:
0.6950 - val_loss: 0.5660 - val_accuracy: 0.8000
Epoch 11/20
10/10 [==============================] - 2s 236ms/step - loss: 0.5749 - accuracy:
0.7500 - val_loss: 0.5514 - val_accuracy: 0.8200
Epoch 12/20
10/10 [==============================] - 1s 131ms/step - loss: 0.5953 - accuracy:
0.7950 - val_loss: 0.5381 - val_accuracy: 0.8300
Epoch 13/20
10/10 [==============================] - 1s 134ms/step - loss: 0.5709 - accuracy:
0.7700 - val_loss: 0.5251 - val_accuracy: 0.8300
Epoch 14/20
10/10 [==============================] - 2s 203ms/step - loss: 0.6213 - accuracy:
0.7600 - val_loss: 0.5127 - val_accuracy: 0.8550
Epoch 15/20
10/10 [==============================] - 2s 213ms/step - loss: 0.5356 - accuracy:
0.8000 - val_loss: 0.5013 - val_accuracy: 0.8600
Epoch 16/20
10/10 [==============================] - 2s 236ms/step - loss: 0.5467 - accuracy:
0.7900 - val_loss: 0.4900 - val_accuracy: 0.8650
Epoch 17/20
10/10 [==============================] - 5s 454ms/step - loss: 0.5103 - accuracy:
0.8250 - val_loss: 0.4793 - val_accuracy: 0.8700
Epoch 18/20
10/10 [==============================] - 2s 236ms/step - loss: 0.5168 - accuracy:
0.8150 - val_loss: 0.4701 - val_accuracy: 0.8700
Epoch 19/20
10/10 [==============================] - 1s 136ms/step - loss: 0.5081 - accuracy:
0.8150 - val_loss: 0.4611 - val_accuracy: 0.8800
Epoch 20/20
10/10 [==============================] - 1s 133ms/step - loss: 0.4756 - accuracy:
```

```
0.8400 - val_loss: 0.4520 - val_accuracy: 0.8800
```

We can display the Accuracy of training and testing data as a graph

In [78]:
```
# Plot training & validation accuracy values
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



As we can see that the highest training accuracy was 84% and highest testing accuracy was 88%. This model is better than CNN and sequential model.

Analysis

For Sequential model the final accuracies for training and testing after 10 epochs were 54% and 53% respectively. The accuracies were low for different epochs other than 10. For CNN model The final accuracies for training and testing after 30 epochs were 96% and 69.5% respectively. The accuracies were low for different epochs other than 30.For Pretrained Model and Transfer Learning the final accuracies for training and testing after 20 epochs were 84% and 88% respectively. It can be seen that with the increasing number of epochs the the accuracy was increasing. It might be that it could become more accurate for higher number of epochs

The CNN Mode gave the highest accuracies for training data which was 96% . Pretrained Model had the highest accuracy for testing data which was 88%.The worst model was the sequential model with 54% and 48.5% for training and testing respectively.

In [80]:
```
!jupyter nbconvert --to html /content/Image_Classification_with_DL.ipynb
```

```
[NbConvertApp] Converting notebook /content/Image_Classification_with_DL.ipynb to h
tml
[NbConvertApp] Writing 2613178 bytes to /content/Image_Classification_with_DL.html
```