```
**Project by:L.CHRISTINA SHERIN
Date:29/12/21**
```

Project by:L.CHRISTINA SHERIN Date:29/12/21

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense,Flatten,Conv2D,MaxPooling2D,Dropout
from tensorflow.keras import layers
from tensorflow.keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

```
#Load the data
from keras.datasets import cifar10
(x_train,y_train),(x_test,y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 11s 0us/step
170508288/170498071 [==============================] - 11s 0us/step
```

```
#Look at the data types of variables
print(type(x_train))
print(type(y_train))
print(type(x_test))
print(type(y_test))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

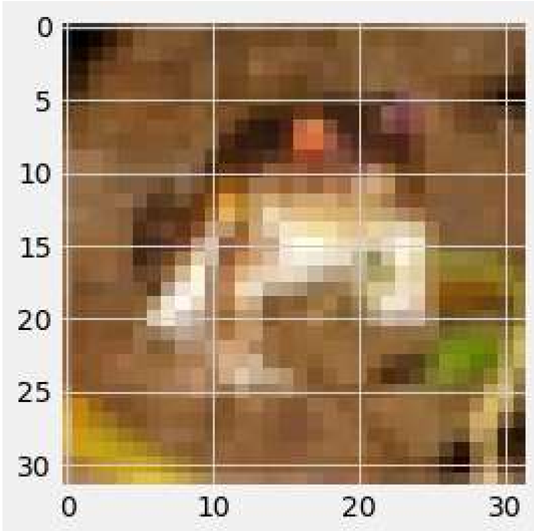
```
#get the shape of the arrays
print('x_train shape:',x_train.shape)
print('y_train shape:',y_train.shape)
print('x_test shape:',x_test.shape)
print('y_test shape:',y_test.shape)
```

```
x_train shape: (50000, 32, 32, 3)
y_train shape: (50000, 1)
x_test shape: (10000, 32, 32, 3)
y_test shape: (10000, 1)
```

```
#take a look at the first image as an array
index=0
x_train[index]
```

```
array([[[ 59,  62,  63],
        [ 43,  46,  45],
        [ 50,  48,  43],
        ...,
        [158, 132, 108],
        [152, 125, 102],
        [148, 124, 103]],

       [[ 16,  20,  20],
        [  0,   0,   0],
        [ 18,   8,   0],
        ...,
        [123,  88,  55],
        [119,  83,  50],
        [122,  87,  57]],

       [[ 25,  24,  21],
        [ 16,   7,   0],
        [ 49,  27,   8],
        ...,
        [118,  84,  50],
        [120,  84,  50],
        [109,  73,  42]],

       ...,

       [[208, 170,  96],
        [201, 153,  34],
        [198, 161,  26],
        ...,
        [160, 133,  70],
        [ 56,  31,   7],
        [ 53,  34,  20]],

       [[180, 139,  96],
        [173, 123,  42],
        [186, 144,  30],
        ...,
        [184, 148,  94],
        [ 97,  62,  34],
        [ 83,  53,  34]],

       [[177, 144, 116],
        [168, 129,  94],
        [179, 142,  87],
        ...,
        [216, 184, 140],
        [151, 118,  84],
        [123,  92,  72]]], dtype=uint8)
```

```
img = plt.imshow(x_train[index])
```



```
#get the image label
print('The image labelis:',y_train[index])
```

```
The image labelis: [6]
```

```
#get the image classification
classification = ['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','tr
#print the image class
print('The image class is:',classification[y_train[index][0]])
```

```
The image class is: frog
```

```
#convert the labels into a set of 10 numbers to input into the neural network
y_train_one_hot = to_categorical(y_train)
y_test_one_hot = to_categorical(y_test)
```

```
#print the new labels
print(y_train_one_hot)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]]
```

```
#print the new label of the current image/picture
print('The one hot label is:',y_train_one_hot[index])
```

```
The one hot label is: [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
```

```
#normalize the pixels to be values between 0 and 1
x_train = x_train / 255
x_test = x_test / 255
```

```
x_train[index]
```

```
array([[[0.23137255, 0.24313725, 0.24705882],
        [0.16862745, 0.18039216, 0.17647059],
        [0.19607843, 0.18823529, 0.16862745],
        ...,
        [0.61960784, 0.51764706, 0.42352941],
        [0.59607843, 0.49019608, 0.4       ],
        [0.58039216, 0.48627451, 0.40392157]],

       [[0.0627451 , 0.07843137, 0.07843137],
        [0.        , 0.        , 0.        ],
        [0.07058824, 0.03137255, 0.        ],
        ...,
        [0.48235294, 0.34509804, 0.21568627],
        [0.46666667, 0.3254902 , 0.19607843],
        [0.47843137, 0.34117647, 0.22352941]],

       [[0.09803922, 0.09411765, 0.08235294],
        [0.0627451 , 0.02745098, 0.        ],
        [0.19215686, 0.10588235, 0.03137255],
        ...,
        [0.4627451 , 0.32941176, 0.19607843],
        [0.47058824, 0.32941176, 0.19607843],
        [0.42745098, 0.28627451, 0.16470588]],

       ...,

       [[0.81568627, 0.66666667, 0.37647059],
        [0.78823529, 0.6       , 0.13333333],
        [0.77647059, 0.63137255, 0.10196078],
        ...,
        [0.62745098, 0.52156863, 0.2745098 ],
        [0.21960784, 0.12156863, 0.02745098],
        [0.20784314, 0.13333333, 0.07843137]],

       [[0.70588235, 0.54509804, 0.37647059],
        [0.67843137, 0.48235294, 0.16470588],
        [0.72941176, 0.56470588, 0.11764706],
        ...,
        [0.72156863, 0.58039216, 0.36862745],
        [0.38039216, 0.24313725, 0.13333333],
        [0.3254902 , 0.20784314, 0.13333333]],

       [[0.69411765, 0.56470588, 0.45490196],
        [0.65882353, 0.50588235, 0.36862745],
        [0.70196078, 0.55686275, 0.34117647],
        ...,
        [0.84705882, 0.72156863, 0.54901961],
```

```
             [0.59215686, 0.4627451 , 0.32941176],
             [0.48235294, 0.36078431, 0.28235294]]]])
```

```
#create the model architecture
model = Sequential()
#add first layer
model.add(Conv2D(32,(5,5),activation='relu',input_shape=(32,32,3)))
#add a pooling layer
model.add(MaxPooling2D(pool_size = (2,2)))
#add another convolution layer
model.add(Conv2D(32,(5,5),activation='relu',input_shape=(32,32,3)))
#add another pooling layer
model.add(MaxPooling2D(pool_size = (2,2)))

#add a flattening layer
model.add(Flatten())

#add a layer with 1000 layers
model.add(Dense(1000,activation='relu'))

#add a drop out layer
model.add(Dropout(0.5))

#add a layer with 500 layers
model.add(Dense(1000,activation='relu'))

#add a drop out layer
model.add(Dropout(0.5))


#add a layer with 250 layers
model.add(Dense(250,activation='relu'))


#add a layer with 10 layers
model.add(Dense(10,activation='softmax'))


#compile the model
model.compile(loss = 'categorical_crossentropy',optimizer='adam',metrics=['accuracy'])


#train the model
hist = model.fit(x_train,y_train_one_hot,batch_size=256,epochs=10,validation_split=0.2)
```

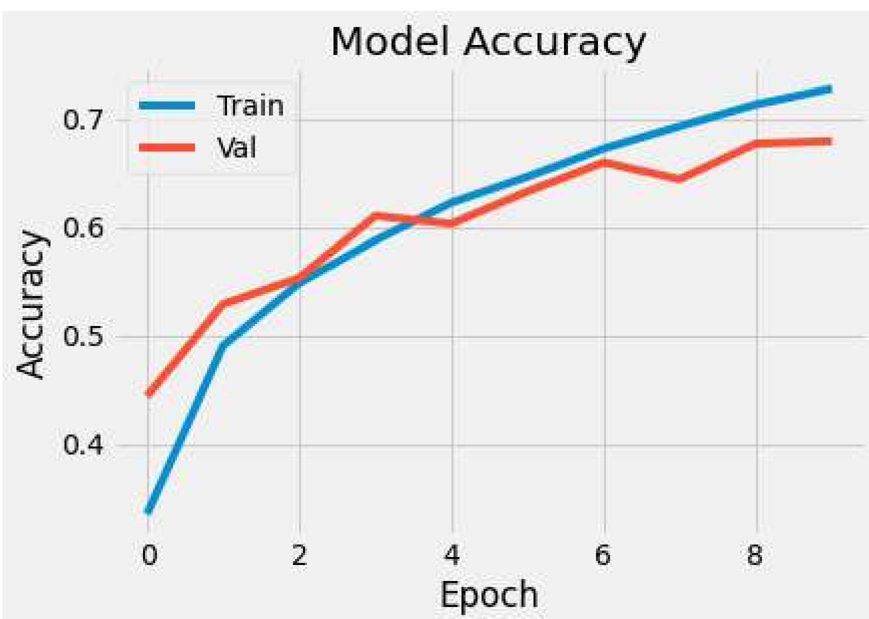
```
Epoch 1/10
157/157 [==============================] - 65s 403ms/step - loss: 1.7846 - accuracy: 0.3
Epoch 2/10
157/157 [==============================] - 64s 407ms/step - loss: 1.4024 - accuracy: 0.4
Epoch 3/10
157/157 [==============================] - 73s 465ms/step - loss: 1.2633 - accuracy: 0.5
Epoch 4/10
157/157 [==============================] - 63s 401ms/step - loss: 1.1575 - accuracy: 0.5
```

```
Epoch 5/10
157/157 [==============================] - 63s 402ms/step - loss: 1.0635 - accuracy: 0.6
Epoch 6/10
157/157 [==============================] - 63s 403ms/step - loss: 0.9999 - accuracy: 0.6
Epoch 7/10
157/157 [==============================] - 63s 403ms/step - loss: 0.9270 - accuracy: 0.6
Epoch 8/10
157/157 [==============================] - 64s 407ms/step - loss: 0.8660 - accuracy: 0.6
Epoch 9/10
157/157 [==============================] - 63s 402ms/step - loss: 0.8060 - accuracy: 0.7
Epoch 10/10
157/157 [==============================] - 63s 403ms/step - loss: 0.7648 - accuracy: 0.7
```

```
model.evaluate(x_test,y_test_one_hot)[1]
```

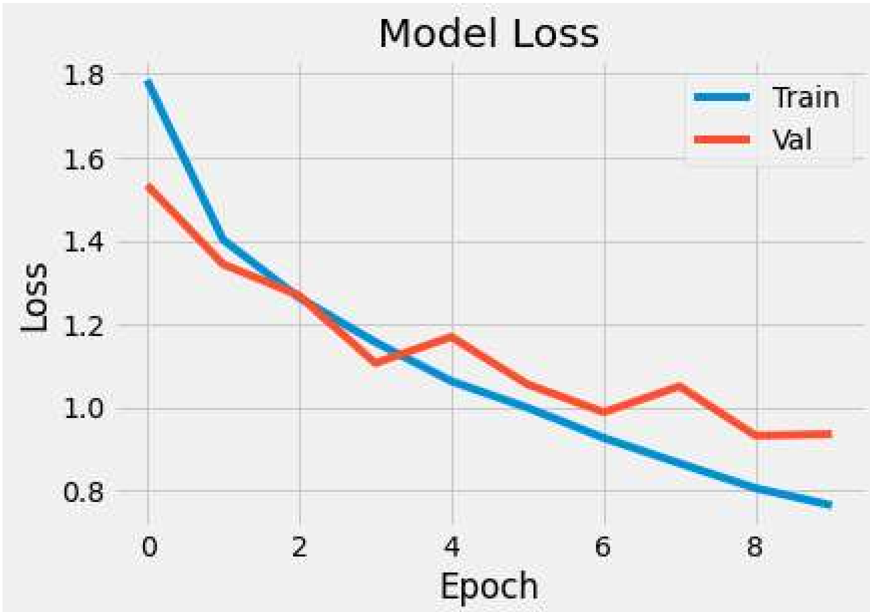
```
313/313 [==============================] - 6s 19ms/step - loss: 0.9494 - accuracy: 0.669
0.6694999933242798
```

```
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train','Val'],loc='upper left')
plt.show()
```




```
#visalise the models loss
```

```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train','Val'],loc='upper right')
plt.show()
```



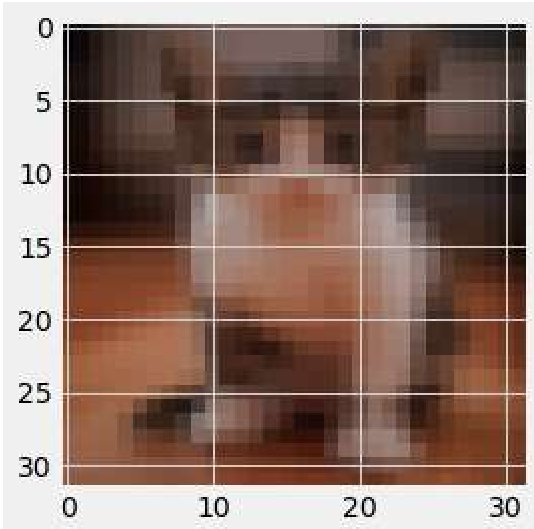

```
#test the model with an exmple
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving k.jpg to k.jpg

```
#show the image
new_image = plt.imread('k.jpg')
img = plt.imshow(new_image)
```

```
#resize the image

from skimage.transform import resize
resized_image = resize(new_image,(32,32,3))
img = plt.imshow(resized_image)
```




```
#get the models prediction
predictions = model.predict(np.array([resized_image]))

#show the predictions
predictions
```

```
array([[0.02222436, 0.00546992, 0.05909045, 0.34055468, 0.0881229 ,
        0.30987534, 0.0596545 , 0.10063497, 0.00726401, 0.00710883]],
      dtype=float32)
```

```
#sort predictions from least to greatest
list_index = [0,1,2,3,4,5,6,7,8,9]
x = predictions
for i in range(10):
  for j in range(10):
    if x[0][list_index[i]] > x[0][list_index[j]]:
      temp = list_index[i]
      list_index[i] = list_index[j]
      list_index[j] = temp


#show the sorted labels in order
print(list_index)
```

```
[3, 5, 7, 4, 6, 2, 0, 8, 9, 1]
```

```
#print the first 5 predictions
for i in range(5):
  print(classification[list_index[i]],':',round(predictions[0][list_index[i]]*100,2),'%')
```

```
cat : 34.06 %
dog : 30.99 %
horse : 10.06 %
deer : 8.81 %
frog : 5.97 %
```