

Submitted by-

NAME: SHAWGAT ASHRAFI

ASTU ROLL NUMBER: 170610107063

SEMESTER: 6TH SEMESTER

COLLEGE: ASSAM ENGINEERING COLLEGE

Topic: Build an Image Classifier to classify images of CIFAR-10 Data

Duration: 2 Weeks

Problem Statement: The dataset stands for the Canadian Institute for Advanced Research (CIFAR). CIFAR Data set is widely used for machine learning and computer vision applications.

- The model should classify the images correctly with respect to its classes as mentioned below.
- CIFAR-10 is a dataset that consists of several images divided into the following 10 (0-9) classes respectively: Airplanes, Cars, Birds, Cats, Deer, Dogs, Frogs, Horses, Ships, Trucks
- The dataset consists of 60,000 32x32 color images, 6,000 images of each class.
- Data Source link: <https://www.cs.toronto.edu/~kriz/cifar.html>

EXECUTION OF THE MODEL:

1. Importing the libraries:

```
Desktop/Image classification - x CIFAR10data_Image_Classific...
localhost:8888/notebooks/Desktop/Image%20classification%20of%20Images%20of%20CIFAR-10%20Data/CIFAR10data_Image_Classification.ipynb
jupyter CIFAR10data_Image_Classification Last Checkpoint: 23 minutes ago (autosaved)
File Edit View Insert Cell Kernel Help Trusted Python 3
In [1]: import numpy as np
import tensorflow as tf
from tensorflow.keras import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import sparse_categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10

In [2]: (x_train, y_train), (x_test, y_test) = cifar10.load_data()

In [3]: x_train.shape[0]
# x_test.shape
Out[3]: 50000

In [4]: #viewing some data
for i in range(10):
    image_index = i
    print(y_train[image_index])
    plt.imshow(x_train[image_index], cmap='Greys')
    plt.show()

[6]
0
5
10
15
20
25
30
```

2. Viewing the data and printing the image shape:

```
Desktop/Image classification - x CIFAR10data_Image_Classific...
localhost:8888/notebooks/Desktop/Image%20classification%20of%20Images%20of%20CIFAR-10%20Data/CIFAR10data_Image_Classification.ipynb
jupyter CIFAR10data_Image_Classification Last Checkpoint: 23 minutes ago (autosaved)
File Edit View Insert Cell Kernel Help Trusted Python 3
In [5]: plt.imshow(x_train[image_index], cmap='Greys')
plt.show()

[6]
0
5
10
15
20
25
30

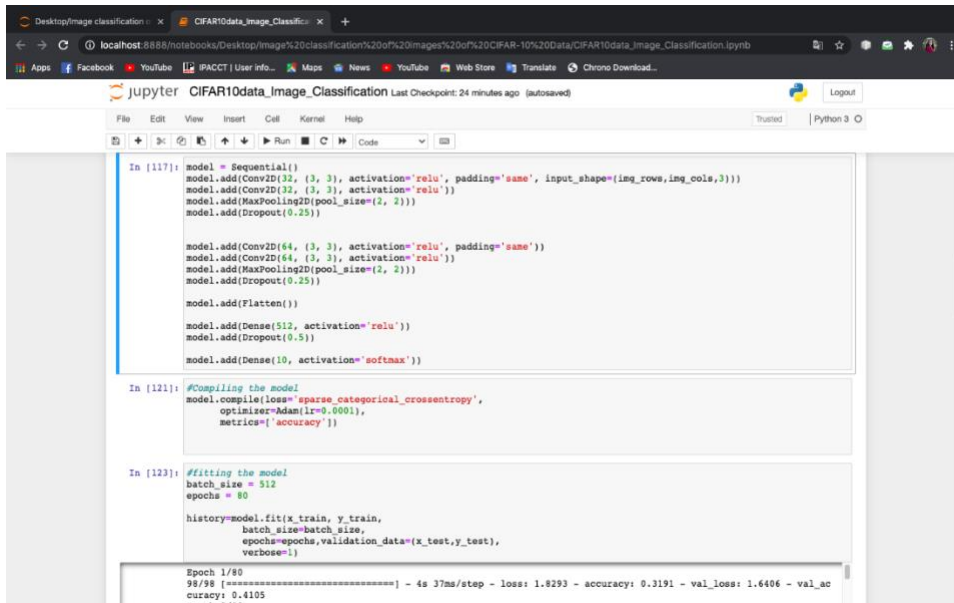
In [6]: img_rows = 32
img_cols = 32
batch_size = 512
img_shape = (img_rows, img_cols, 3)

# x_train = x_train.reshape(x_train.shape[0], img_cols, img_rows, 1)
# x_test = x_test.reshape(x_test.shape[0], img_cols, img_rows, 1)

print('x_train shape: {}'.format(x_train.shape))
print('x_test shape: {}'.format(x_test.shape))
print(img_shape)

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

3. Compiling the model:



The screenshot shows a Jupyter Notebook with three code cells. The first cell defines a sequential model with two convolutional layers, two max pooling layers, two dropout layers, a flatten layer, a dense layer, and a softmax layer. The second cell compiles the model with sparse categorical crossentropy loss, Adam optimizer, and accuracy metric. The third cell fits the model for 80 epochs with a batch size of 512. The output shows the progress of the first epoch.

```
In [117]: model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(img_rows, img_cols, 3)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

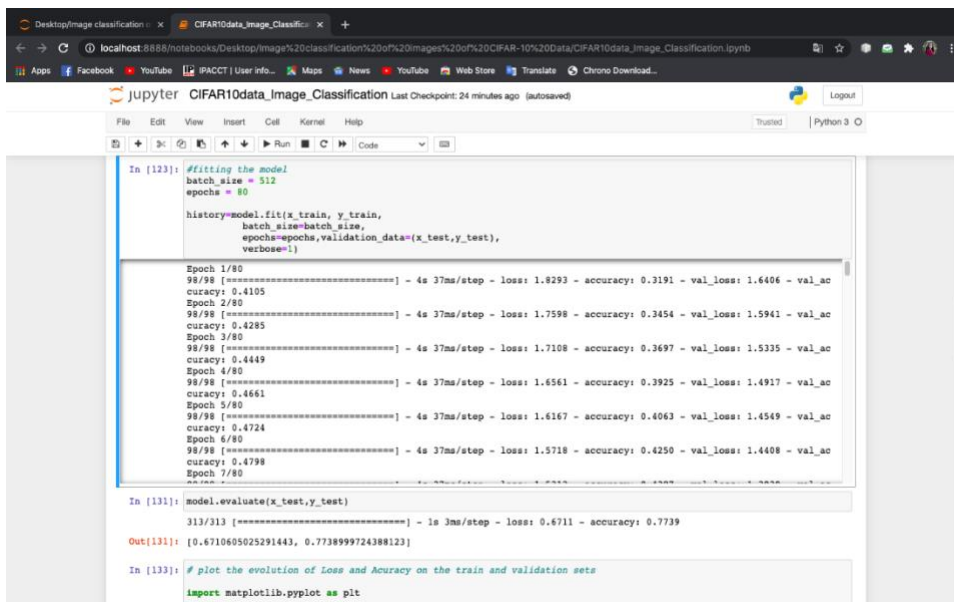
In [121]: #Compiling the model
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=Adam(lr=0.0001),
              metrics=['accuracy'])

In [123]: #fitting the model
batch_size = 512
epochs = 80

history=model.fit(x_train, y_train,
                  batch_size=batch_size,
                  epochs=epochs, validation_data=(x_test, y_test),
                  verbose=1)

Epoch 1/80
98/98 [=====] - 4s 37ms/step - loss: 1.8293 - accuracy: 0.3191 - val_loss: 1.6406 - val_ac
curacy: 0.4105
```

4. Fitting the model:



The screenshot shows the continuation of the Jupyter Notebook. The third cell from the previous image is repeated, showing the full 80 epochs of training. The output displays the progress for each epoch, including loss, accuracy, and validation loss/accuracy. The final cell evaluates the model on the test set, and the last cell imports matplotlib for plotting.

```
In [123]: #fitting the model
batch_size = 512
epochs = 80

history=model.fit(x_train, y_train,
                  batch_size=batch_size,
                  epochs=epochs, validation_data=(x_test, y_test),
                  verbose=1)

Epoch 1/80
98/98 [=====] - 4s 37ms/step - loss: 1.8293 - accuracy: 0.3191 - val_loss: 1.6406 - val_ac
curacy: 0.4105
Epoch 2/80
98/98 [=====] - 4s 37ms/step - loss: 1.7598 - accuracy: 0.3454 - val_loss: 1.5941 - val_ac
curacy: 0.4285
Epoch 3/80
98/98 [=====] - 4s 37ms/step - loss: 1.7108 - accuracy: 0.3697 - val_loss: 1.5335 - val_ac
curacy: 0.4449
Epoch 4/80
98/98 [=====] - 4s 37ms/step - loss: 1.6561 - accuracy: 0.3925 - val_loss: 1.4917 - val_ac
curacy: 0.4661
Epoch 5/80
98/98 [=====] - 4s 37ms/step - loss: 1.6167 - accuracy: 0.4063 - val_loss: 1.4549 - val_ac
curacy: 0.4724
Epoch 6/80
98/98 [=====] - 4s 37ms/step - loss: 1.5718 - accuracy: 0.4250 - val_loss: 1.4408 - val_ac
curacy: 0.4798
Epoch 7/80
98/98 [=====] - 4s 37ms/step - loss: 1.5318 - accuracy: 0.4425 - val_loss: 1.4288 - val_ac
curacy: 0.4855
Epoch 8/80
98/98 [=====] - 4s 37ms/step - loss: 1.4961 - accuracy: 0.4568 - val_loss: 1.4181 - val_ac
curacy: 0.4908
Epoch 9/80
98/98 [=====] - 4s 37ms/step - loss: 1.4641 - accuracy: 0.4697 - val_loss: 1.4088 - val_ac
curacy: 0.4958
Epoch 10/80
98/98 [=====] - 4s 37ms/step - loss: 1.4354 - accuracy: 0.4811 - val_loss: 1.4008 - val_ac
curacy: 0.5005
Epoch 11/80
98/98 [=====] - 4s 37ms/step - loss: 1.4101 - accuracy: 0.4911 - val_loss: 1.3939 - val_ac
curacy: 0.5048
Epoch 12/80
98/98 [=====] - 4s 37ms/step - loss: 1.3878 - accuracy: 0.5000 - val_loss: 1.3881 - val_ac
curacy: 0.5087
Epoch 13/80
98/98 [=====] - 4s 37ms/step - loss: 1.3681 - accuracy: 0.5078 - val_loss: 1.3833 - val_ac
curacy: 0.5122
Epoch 14/80
98/98 [=====] - 4s 37ms/step - loss: 1.3507 - accuracy: 0.5146 - val_loss: 1.3794 - val_ac
curacy: 0.5154
Epoch 15/80
98/98 [=====] - 4s 37ms/step - loss: 1.3354 - accuracy: 0.5204 - val_loss: 1.3763 - val_ac
curacy: 0.5183
Epoch 16/80
98/98 [=====] - 4s 37ms/step - loss: 1.3221 - accuracy: 0.5252 - val_loss: 1.3739 - val_ac
curacy: 0.5209
Epoch 17/80
98/98 [=====] - 4s 37ms/step - loss: 1.3106 - accuracy: 0.5291 - val_loss: 1.3721 - val_ac
curacy: 0.5232
Epoch 18/80
98/98 [=====] - 4s 37ms/step - loss: 1.3008 - accuracy: 0.5321 - val_loss: 1.3708 - val_ac
curacy: 0.5252
Epoch 19/80
98/98 [=====] - 4s 37ms/step - loss: 1.2926 - accuracy: 0.5343 - val_loss: 1.3700 - val_ac
curacy: 0.5269
Epoch 20/80
98/98 [=====] - 4s 37ms/step - loss: 1.2850 - accuracy: 0.5357 - val_loss: 1.3696 - val_ac
curacy: 0.5283
Epoch 21/80
98/98 [=====] - 4s 37ms/step - loss: 1.2781 - accuracy: 0.5373 - val_loss: 1.3694 - val_ac
curacy: 0.5294
Epoch 22/80
98/98 [=====] - 4s 37ms/step - loss: 1.2718 - accuracy: 0.5381 - val_loss: 1.3694 - val_ac
curacy: 0.5303
Epoch 23/80
98/98 [=====] - 4s 37ms/step - loss: 1.2661 - accuracy: 0.5389 - val_loss: 1.3695 - val_ac
curacy: 0.5310
Epoch 24/80
98/98 [=====] - 4s 37ms/step - loss: 1.2609 - accuracy: 0.5396 - val_loss: 1.3697 - val_ac
curacy: 0.5316
Epoch 25/80
98/98 [=====] - 4s 37ms/step - loss: 1.2562 - accuracy: 0.5402 - val_loss: 1.3699 - val_ac
curacy: 0.5321
Epoch 26/80
98/98 [=====] - 4s 37ms/step - loss: 1.2519 - accuracy: 0.5408 - val_loss: 1.3702 - val_ac
curacy: 0.5325
Epoch 27/80
98/98 [=====] - 4s 37ms/step - loss: 1.2480 - accuracy: 0.5413 - val_loss: 1.3705 - val_ac
curacy: 0.5328
Epoch 28/80
98/98 [=====] - 4s 37ms/step - loss: 1.2444 - accuracy: 0.5418 - val_loss: 1.3708 - val_ac
curacy: 0.5331
Epoch 29/80
98/98 [=====] - 4s 37ms/step - loss: 1.2411 - accuracy: 0.5422 - val_loss: 1.3711 - val_ac
curacy: 0.5334
Epoch 30/80
98/98 [=====] - 4s 37ms/step - loss: 1.2381 - accuracy: 0.5426 - val_loss: 1.3714 - val_ac
curacy: 0.5337
Epoch 31/80
98/98 [=====] - 4s 37ms/step - loss: 1.2353 - accuracy: 0.5429 - val_loss: 1.3717 - val_ac
curacy: 0.5339
Epoch 32/80
98/98 [=====] - 4s 37ms/step - loss: 1.2327 - accuracy: 0.5432 - val_loss: 1.3720 - val_ac
curacy: 0.5341
Epoch 33/80
98/98 [=====] - 4s 37ms/step - loss: 1.2303 - accuracy: 0.5435 - val_loss: 1.3723 - val_ac
curacy: 0.5343
Epoch 34/80
98/98 [=====] - 4s 37ms/step - loss: 1.2280 - accuracy: 0.5437 - val_loss: 1.3726 - val_ac
curacy: 0.5345
Epoch 35/80
98/98 [=====] - 4s 37ms/step - loss: 1.2258 - accuracy: 0.5439 - val_loss: 1.3729 - val_ac
curacy: 0.5347
Epoch 36/80
98/98 [=====] - 4s 37ms/step - loss: 1.2237 - accuracy: 0.5441 - val_loss: 1.3732 - val_ac
curacy: 0.5348
Epoch 37/80
98/98 [=====] - 4s 37ms/step - loss: 1.2217 - accuracy: 0.5443 - val_loss: 1.3735 - val_ac
curacy: 0.5349
Epoch 38/80
98/98 [=====] - 4s 37ms/step - loss: 1.2198 - accuracy: 0.5445 - val_loss: 1.3738 - val_ac
curacy: 0.5350
Epoch 39/80
98/98 [=====] - 4s 37ms/step - loss: 1.2180 - accuracy: 0.5446 - val_loss: 1.3741 - val_ac
curacy: 0.5351
Epoch 40/80
98/98 [=====] - 4s 37ms/step - loss: 1.2163 - accuracy: 0.5447 - val_loss: 1.3744 - val_ac
curacy: 0.5352
Epoch 41/80
98/98 [=====] - 4s 37ms/step - loss: 1.2147 - accuracy: 0.5448 - val_loss: 1.3747 - val_ac
curacy: 0.5353
Epoch 42/80
98/98 [=====] - 4s 37ms/step - loss: 1.2131 - accuracy: 0.5449 - val_loss: 1.3750 - val_ac
curacy: 0.5354
Epoch 43/80
98/98 [=====] - 4s 37ms/step - loss: 1.2116 - accuracy: 0.5450 - val_loss: 1.3753 - val_ac
curacy: 0.5355
Epoch 44/80
98/98 [=====] - 4s 37ms/step - loss: 1.2101 - accuracy: 0.5451 - val_loss: 1.3756 - val_ac
curacy: 0.5356
Epoch 45/80
98/98 [=====] - 4s 37ms/step - loss: 1.2087 - accuracy: 0.5452 - val_loss: 1.3759 - val_ac
curacy: 0.5357
Epoch 46/80
98/98 [=====] - 4s 37ms/step - loss: 1.2073 - accuracy: 0.5453 - val_loss: 1.3762 - val_ac
curacy: 0.5358
Epoch 47/80
98/98 [=====] - 4s 37ms/step - loss: 1.2060 - accuracy: 0.5454 - val_loss: 1.3765 - val_ac
curacy: 0.5359
Epoch 48/80
98/98 [=====] - 4s 37ms/step - loss: 1.2047 - accuracy: 0.5455 - val_loss: 1.3768 - val_ac
curacy: 0.5360
Epoch 49/80
98/98 [=====] - 4s 37ms/step - loss: 1.2035 - accuracy: 0.5456 - val_loss: 1.3771 - val_ac
curacy: 0.5361
Epoch 50/80
98/98 [=====] - 4s 37ms/step - loss: 1.2023 - accuracy: 0.5457 - val_loss: 1.3774 - val_ac
curacy: 0.5362
Epoch 51/80
98/98 [=====] - 4s 37ms/step - loss: 1.2011 - accuracy: 0.5458 - val_loss: 1.3777 - val_ac
curacy: 0.5363
Epoch 52/80
98/98 [=====] - 4s 37ms/step - loss: 1.2000 - accuracy: 0.5459 - val_loss: 1.3780 - val_ac
curacy: 0.5364
Epoch 53/80
98/98 [=====] - 4s 37ms/step - loss: 1.1989 - accuracy: 0.5460 - val_loss: 1.3783 - val_ac
curacy: 0.5365
Epoch 54/80
98/98 [=====] - 4s 37ms/step - loss: 1.1978 - accuracy: 0.5461 - val_loss: 1.3786 - val_ac
curacy: 0.5366
Epoch 55/80
98/98 [=====] - 4s 37ms/step - loss: 1.1968 - accuracy: 0.5462 - val_loss: 1.3789 - val_ac
curacy: 0.5367
Epoch 56/80
98/98 [=====] - 4s 37ms/step - loss: 1.1958 - accuracy: 0.5463 - val_loss: 1.3792 - val_ac
curacy: 0.5368
Epoch 57/80
98/98 [=====] - 4s 37ms/step - loss: 1.1948 - accuracy: 0.5464 - val_loss: 1.3795 - val_ac
curacy: 0.5369
Epoch 58/80
98/98 [=====] - 4s 37ms/step - loss: 1.1939 - accuracy: 0.5465 - val_loss: 1.3798 - val_ac
curacy: 0.5370
Epoch 59/80
98/98 [=====] - 4s 37ms/step - loss: 1.1930 - accuracy: 0.5466 - val_loss: 1.3801 - val_ac
curacy: 0.5371
Epoch 60/80
98/98 [=====] - 4s 37ms/step - loss: 1.1921 - accuracy: 0.5467 - val_loss: 1.3804 - val_ac
curacy: 0.5372
Epoch 61/80
98/98 [=====] - 4s 37ms/step - loss: 1.1912 - accuracy: 0.5468 - val_loss: 1.3807 - val_ac
curacy: 0.5373
Epoch 62/80
98/98 [=====] - 4s 37ms/step - loss: 1.1903 - accuracy: 0.5469 - val_loss: 1.3810 - val_ac
curacy: 0.5374
Epoch 63/80
98/98 [=====] - 4s 37ms/step - loss: 1.1895 - accuracy: 0.5470 - val_loss: 1.3813 - val_ac
curacy: 0.5375
Epoch 64/80
98/98 [=====] - 4s 37ms/step - loss: 1.1887 - accuracy: 0.5471 - val_loss: 1.3816 - val_ac
curacy: 0.5376
Epoch 65/80
98/98 [=====] - 4s 37ms/step - loss: 1.1879 - accuracy: 0.5472 - val_loss: 1.3819 - val_ac
curacy: 0.5377
Epoch 66/80
98/98 [=====] - 4s 37ms/step - loss: 1.1871 - accuracy: 0.5473 - val_loss: 1.3822 - val_ac
curacy: 0.5378
Epoch 67/80
98/98 [=====] - 4s 37ms/step - loss: 1.1863 - accuracy: 0.5474 - val_loss: 1.3825 - val_ac
curacy: 0.5379
Epoch 68/80
98/98 [=====] - 4s 37ms/step - loss: 1.1855 - accuracy: 0.5475 - val_loss: 1.3828 - val_ac
curacy: 0.5380
Epoch 69/80
98/98 [=====] - 4s 37ms/step - loss: 1.1848 - accuracy: 0.5476 - val_loss: 1.3831 - val_ac
curacy: 0.5381
Epoch 70/80
98/98 [=====] - 4s 37ms/step - loss: 1.1840 - accuracy: 0.5477 - val_loss: 1.3834 - val_ac
curacy: 0.5382
Epoch 71/80
98/98 [=====] - 4s 37ms/step - loss: 1.1833 - accuracy: 0.5478 - val_loss: 1.3837 - val_ac
curacy: 0.5383
Epoch 72/80
98/98 [=====] - 4s 37ms/step - loss: 1.1825 - accuracy: 0.5479 - val_loss: 1.3840 - val_ac
curacy: 0.5384
Epoch 73/80
98/98 [=====] - 4s 37ms/step - loss: 1.1818 - accuracy: 0.5480 - val_loss: 1.3843 - val_ac
curacy: 0.5385
Epoch 74/80
98/98 [=====] - 4s 37ms/step - loss: 1.1811 - accuracy: 0.5481 - val_loss: 1.3846 - val_ac
curacy: 0.5386
Epoch 75/80
98/98 [=====] - 4s 37ms/step - loss: 1.1804 - accuracy: 0.5482 - val_loss: 1.3849 - val_ac
curacy: 0.5387
Epoch 76/80
98/98 [=====] - 4s 37ms/step - loss: 1.1797 - accuracy: 0.5483 - val_loss: 1.3852 - val_ac
curacy: 0.5388
Epoch 77/80
98/98 [=====] - 4s 37ms/step - loss: 1.1790 - accuracy: 0.5484 - val_loss: 1.3855 - val_ac
curacy: 0.5389
Epoch 78/80
98/98 [=====] - 4s 37ms/step - loss: 1.1783 - accuracy: 0.5485 - val_loss: 1.3858 - val_ac
curacy: 0.5390
Epoch 79/80
98/98 [=====] - 4s 37ms/step - loss: 1.1776 - accuracy: 0.5486 - val_loss: 1.3861 - val_ac
curacy: 0.5391
Epoch 80/80
98/98 [=====] - 4s 37ms/step - loss: 1.1770 - accuracy: 0.5487 - val_loss: 1.3864 - val_ac
curacy: 0.5392

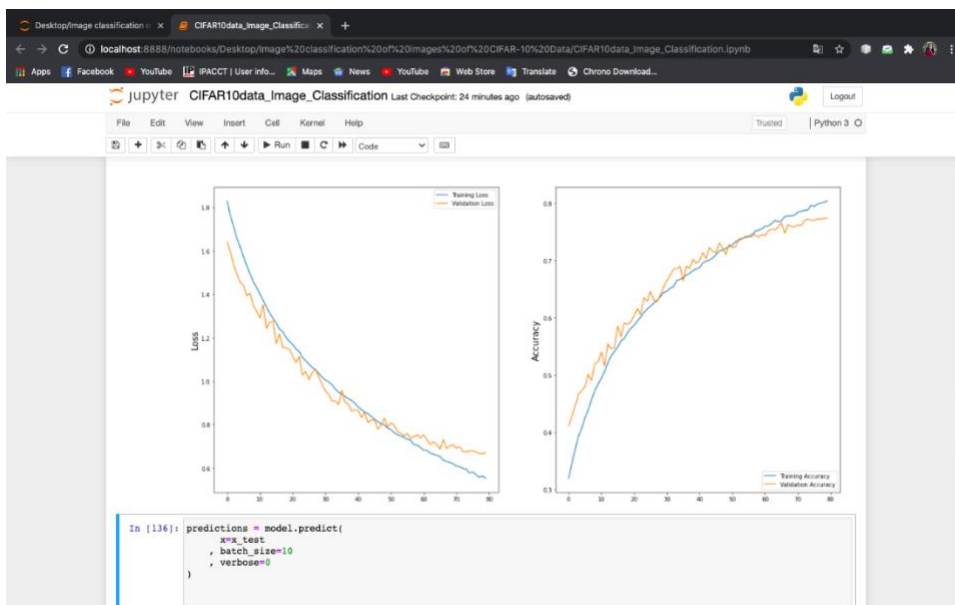
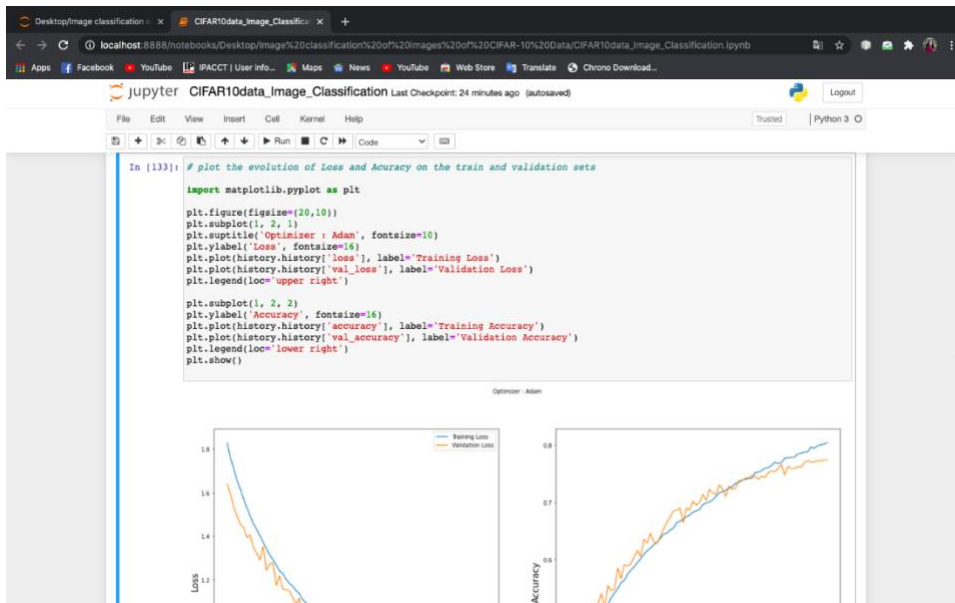
In [131]: model.evaluate(x_test, y_test)

313/313 [=====] - 1s 3ms/step - loss: 0.6711 - accuracy: 0.7739

Out[131]: [0.6710605025291443, 0.7738999724388123]

In [133]: # plot the evolution of Loss and Accuracy on the train and validation sets
import matplotlib.pyplot as plt
```

5. Plotting the evolution of Loss and Accuracy on the train and validation sets:



6. Model predictions:

```
DesktopImage classification - x CIFAR10data_image_Classific... +
localhost:8888/notebooks/DesktopImage%20classification%20of%20Images%20of%20CIFAR-10%20Data/CIFAR10data_image_Classification.ipynb
jupyter CIFAR10data_image_Classification Last Checkpoint: 24 minutes ago (autosaved)
Python 3

In [136]: predictions = model.predict(
          x=x_test
          , batch_size=10
          , verbose=0
        )

In [137]: for i in predictions:
          print(i)

In [140]: labels_pred = np.argmax(predictions,axis=1)
          print(labels_pred)

          [3 8 8 ... 5 0 7]

In [145]: correct = (labels_pred == y_test)
          print(correct)

          [[ True False False ... False False False]
          [False True  True ... False False False]
          [False True  True ... False False False]
          ...
          [False False False ...  True False False]
          [False False False ... False False False]
          [False False False ... False False  True]]

In [160]: 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(x_test[i], cmap=plt.cm.binary)
              plt.xlabel(labels_pred[i])
```

FINAL OUTPUT-

