ASSIGNMENT-7- Image Classification

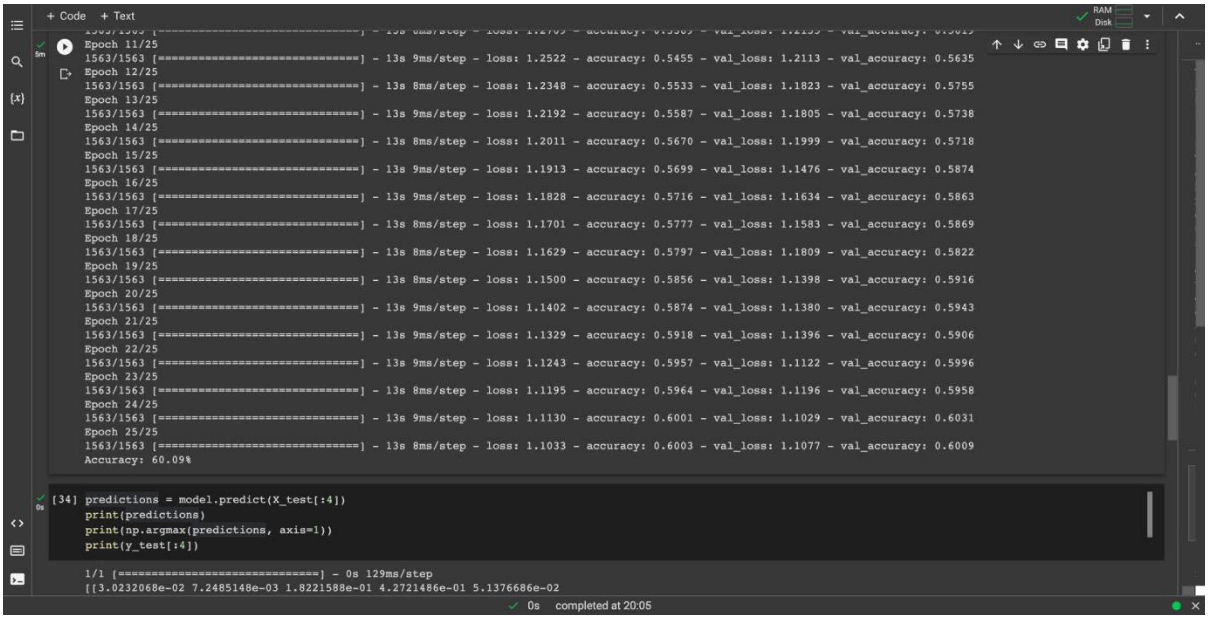
Samudrala Bindu- 700744928

Github: https://github.com/Samudralabindu/ICP7

Video link:
https://drive.google.com/file/d/1nm1XBu7dsunXJ9QE4eQnQeFTLm_rXKT2/view?usp=drive_link

```
importnumpyasnp
fromkeras.datasetsimport cifar10
fromkeras.modelsimport Sequential
fromkeras.layersimport Dense, Dropout, Flatten, Conv2D, MaxPooling2D
fromkeras.constraintsimportmaxnorm
fromkeras.optimizersimport SGD
fromkeras.utilsimportnp_utils
# from keras import backend as K
# K.tensorflow_backend.set_image_dim_ordering('th')
# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# convert from int to float and normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]

# transpose the dimensions of the input data
X_train = np.transpose(X_train, (0, 3, 1, 2))
X_test = np.transpose(X_test, (0, 3, 1, 2))

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), padding='same',
activation='relu', kernel_constraint=maxnorm(3)))
```

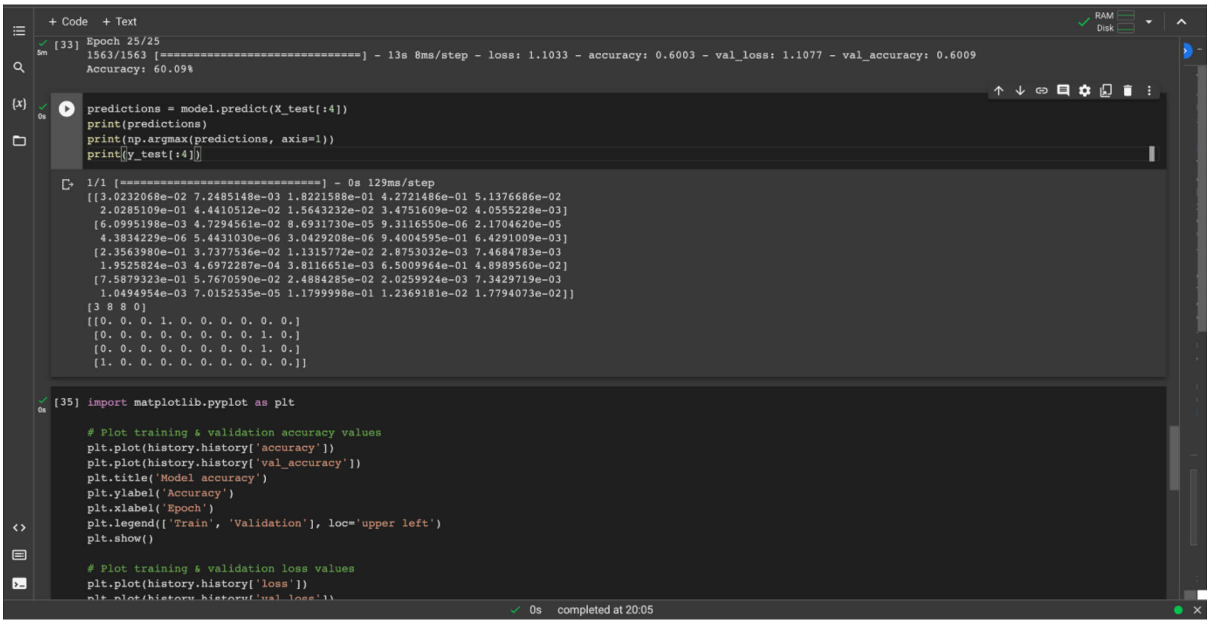
```
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu',
kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same',
kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(Conv2D(128, (3, 3), padding='same', activation='relu',
kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same',
kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
# Fit the model
history=model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```



The Accuracy Score obtained by the model is slightly changed.Initially the accuracy score is around 59.87% and for after the modifications and adding the layers the accuracy is slightly changed.

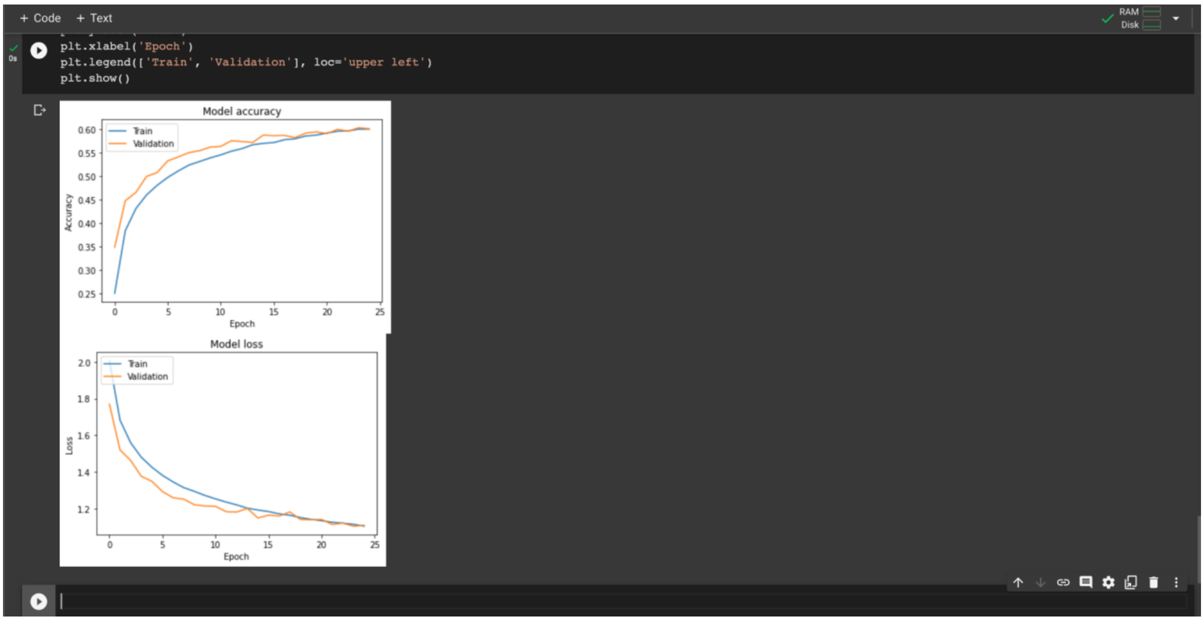
2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly.

```
predictions = model.predict(X_test[:4])
print(predictions)
print(np.argmax(predictions, axis=1))
print(y_test[:4])
```



3. Visualize Loss and Accuracy using the history object

```
importmatplotlib.pyplotasplt

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

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



The CIFAR-10 dataset, which is a well-known dataset of images separated into 10 classes, is loaded using the code above. Training sets and testing sets are created from the data. The input pictures are normalized between 0 and 1 and translated from integers to floats. One-hot encoding is used for the output labels.

The dimensions of the input data are then transformed to fit the convolutional neural network's anticipated input shape (CNN).

The Keras Sequential API is then used to define the CNN model. It is made up of dense layers with ReLU activation, dropout layers to avoid overfitting, and several convolutional layers with ReLU activation. categorize the photos into the 10 classes. classify the images into the 10 classes. classify the image into the 10 classes. The model is built using categorical cross-entropy as the loss function, stochastic gradient descent (SGD) as the optimizer, and accuracy as the training-phase performance indicator.32 batches are used during the 25 epochs of model training. model is evaluated on the test set and the accuracy is printed.

The model is then applied to the first 4 photos in the test set to make predictions. To determine whether the model has predicted accurately, the predicted labels are compared to the actual labels. Lastly, the history object returned by the fit() method is used to illustrate the loss and accuracy.