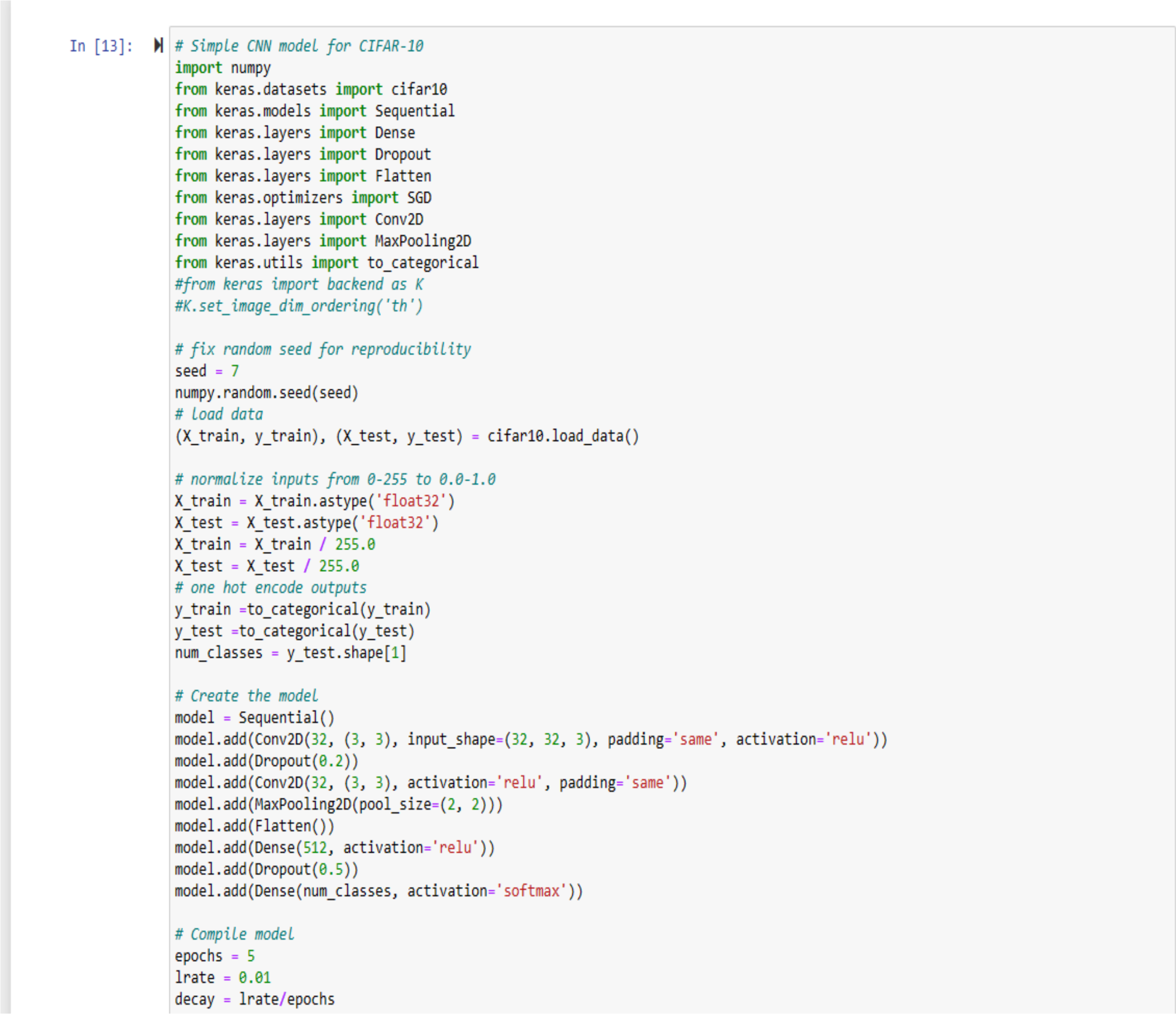
Assignment-7 DEEKSHITH ATHMAKUR 700743388

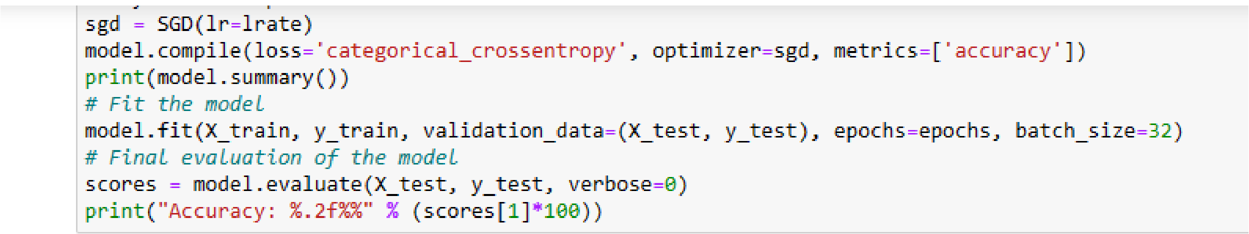
Neural Networks and Deep learning

GitHub link: https://github.com/DEEKSHITH-ATHMAKUR/ICP7

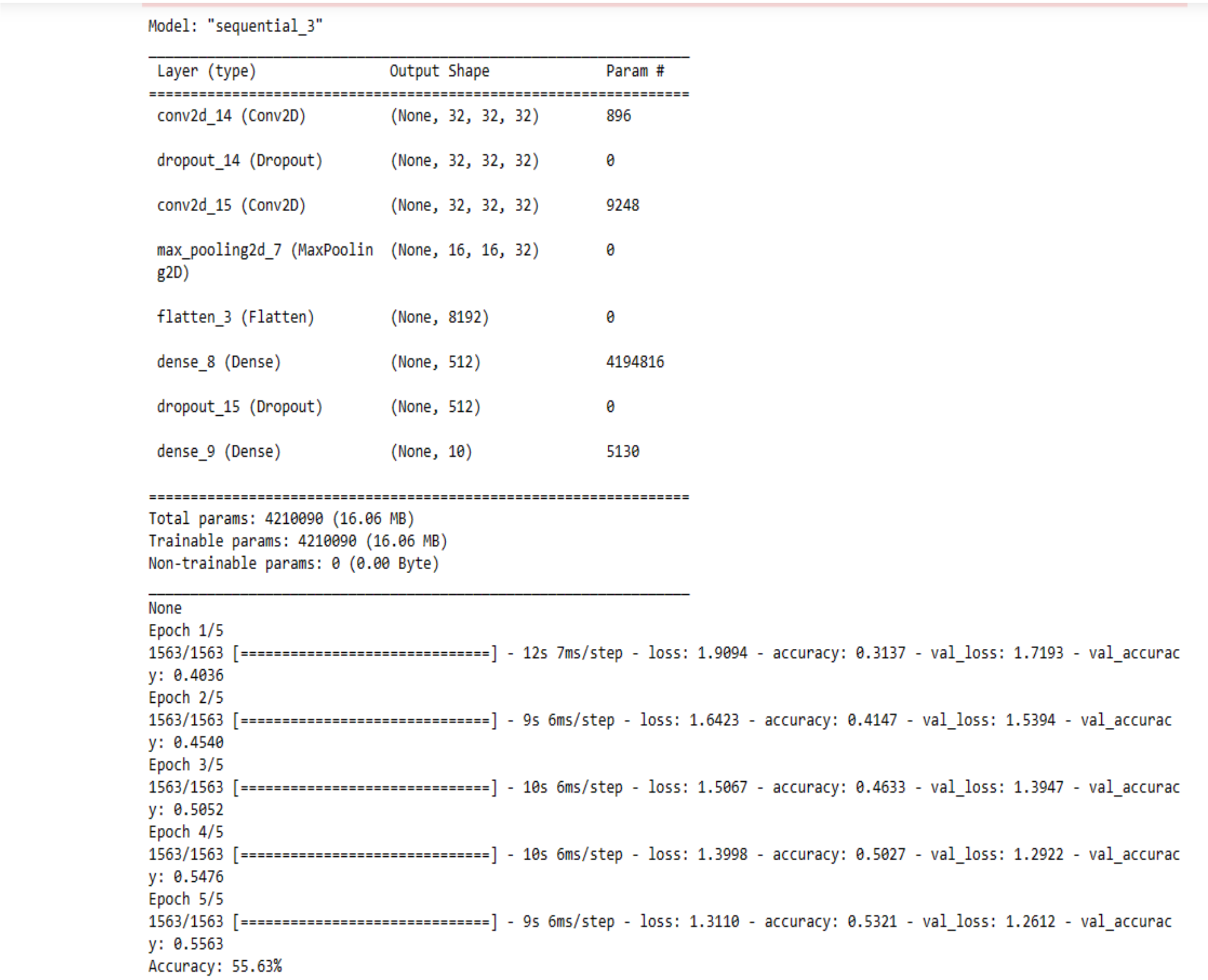


In [13]:

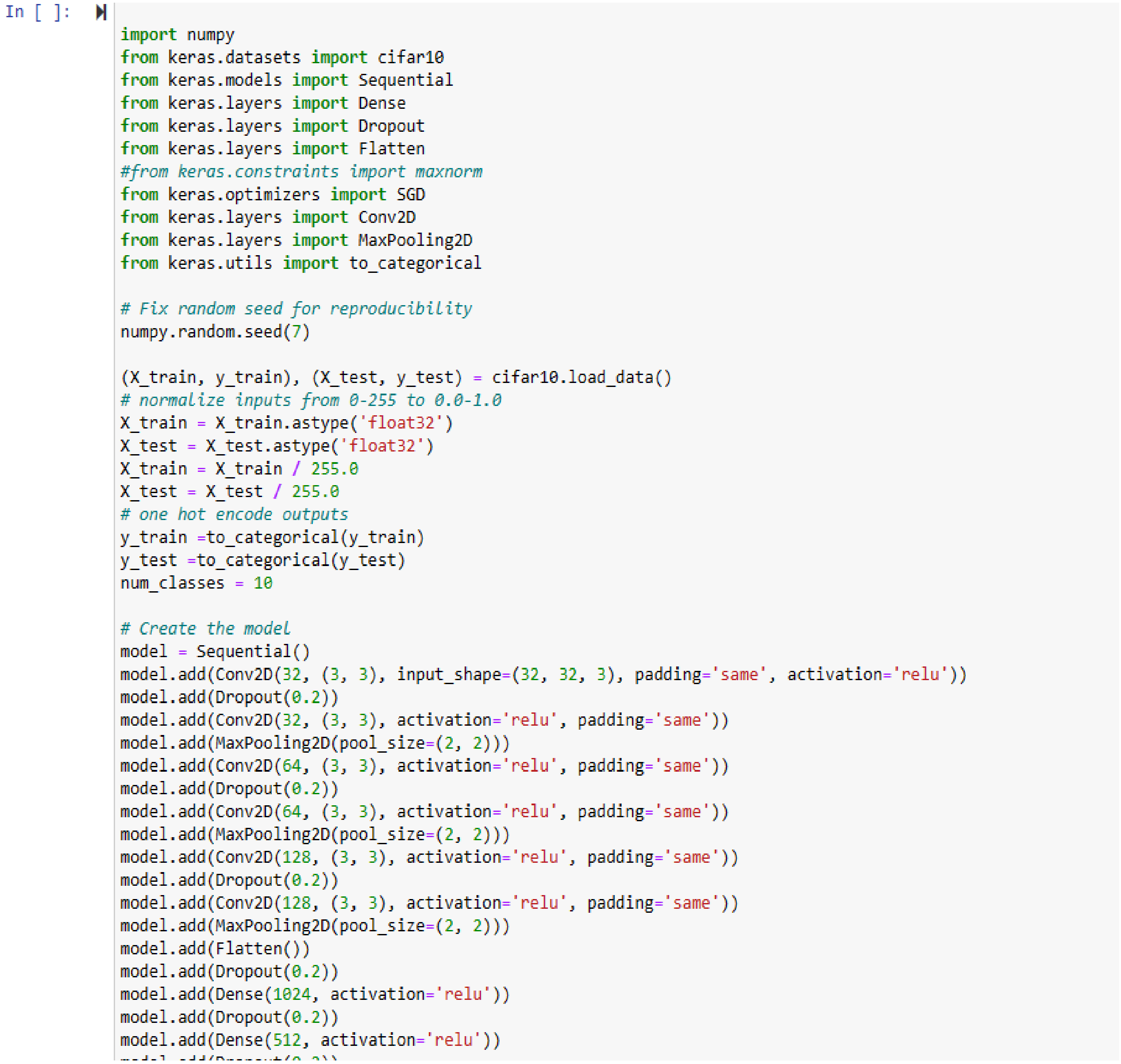
```
# Simple CNN model for CIFAR-10
import numpy
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.optimizers import SGD
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.utils import to_categorical
#from keras import backend as K
#K.set_image_dim_ordering('th')

# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
# one hot encode outputs
y_train =to_categorical(y_train)
y_test =to_categorical(y_test)
num_classes = y_test.shape[1]

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu'))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 5
lrate = 0.01
decay = lrate/epochs
```



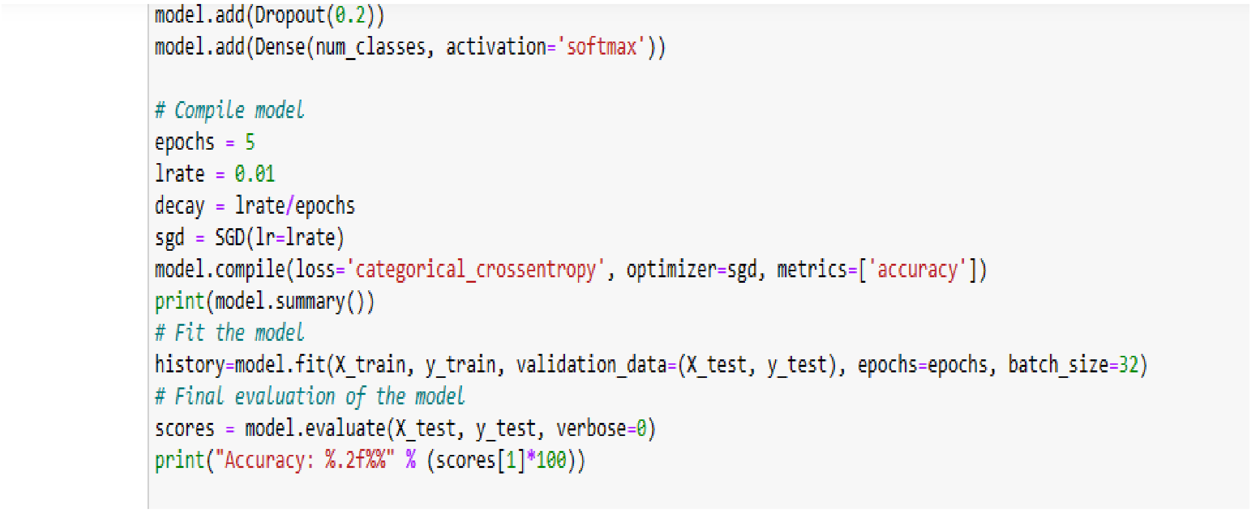
```
sgd = SGD(lr=lrate)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test,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



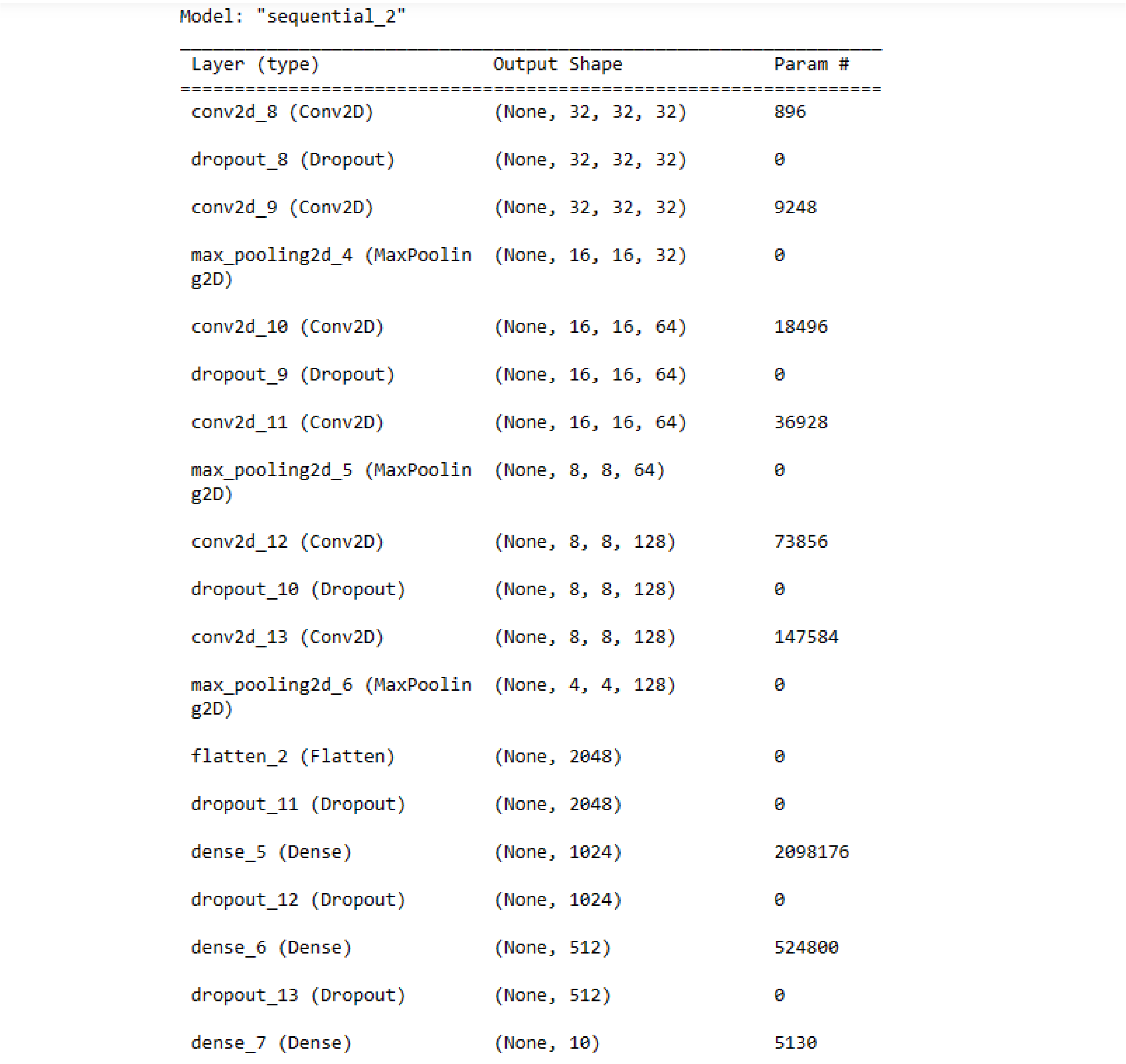
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_14 (Conv2D)          (None, 32, 32, 32)        896       
                                                                 
 dropout_14 (Dropout)        (None, 32, 32, 32)        0         
                                                                 
 conv2d_15 (Conv2D)          (None, 32, 32, 32)        9248      
                                                                 
 max_pooling2d_7 (MaxPoolin  (None, 16, 16, 32)        0         
 g2D)                                                            
                                                                 
 flatten_3 (Flatten)         (None, 8192)              0         
                                                                 
 dense_8 (Dense)             (None, 512)               4194816   
                                                                 
 dropout_15 (Dropout)        (None, 512)               0         
                                                                 
 dense_9 (Dense)             (None, 10)                5130      
                                                                 
=================================================================
Total params: 4210090 (16.06 MB)
Trainable params: 4210090 (16.06 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
Epoch 1/5
1563/1563 [==============================] - 12s 7ms/step - loss: 1.9094 - accuracy: 0.3137 - val_loss: 1.7193 - val_accurac
y: 0.4036
Epoch 2/5
1563/1563 [==============================] - 9s 6ms/step - loss: 1.6423 - accuracy: 0.4147 - val_loss: 1.5394 - val_accurac
y: 0.4540
Epoch 3/5
1563/1563 [==============================] - 10s 6ms/step - loss: 1.5067 - accuracy: 0.4633 - val_loss: 1.3947 - val_accurac
y: 0.5052
Epoch 4/5
1563/1563 [==============================] - 10s 6ms/step - loss: 1.3998 - accuracy: 0.5027 - val_loss: 1.2922 - val_accurac
y: 0.5476
Epoch 5/5
1563/1563 [==============================] - 9s 6ms/step - loss: 1.3110 - accuracy: 0.5321 - val_loss: 1.2612 - val_accurac
y: 0.5563
Accuracy: 55.63%
```



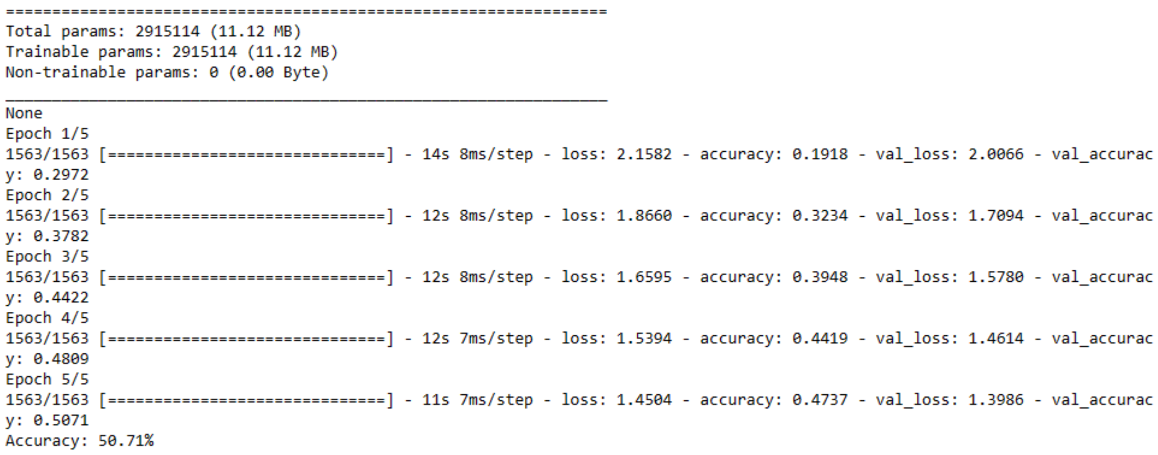
```
In [ ]:
import numpy
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
#from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.utils import to_categorical

# Fix random seed for reproducibility
numpy.random.seed(7)

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
# normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
# one hot encode outputs
y_train =to_categorical(y_train)
y_test =to_categorical(y_test)
num_classes = 10

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu'))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
```



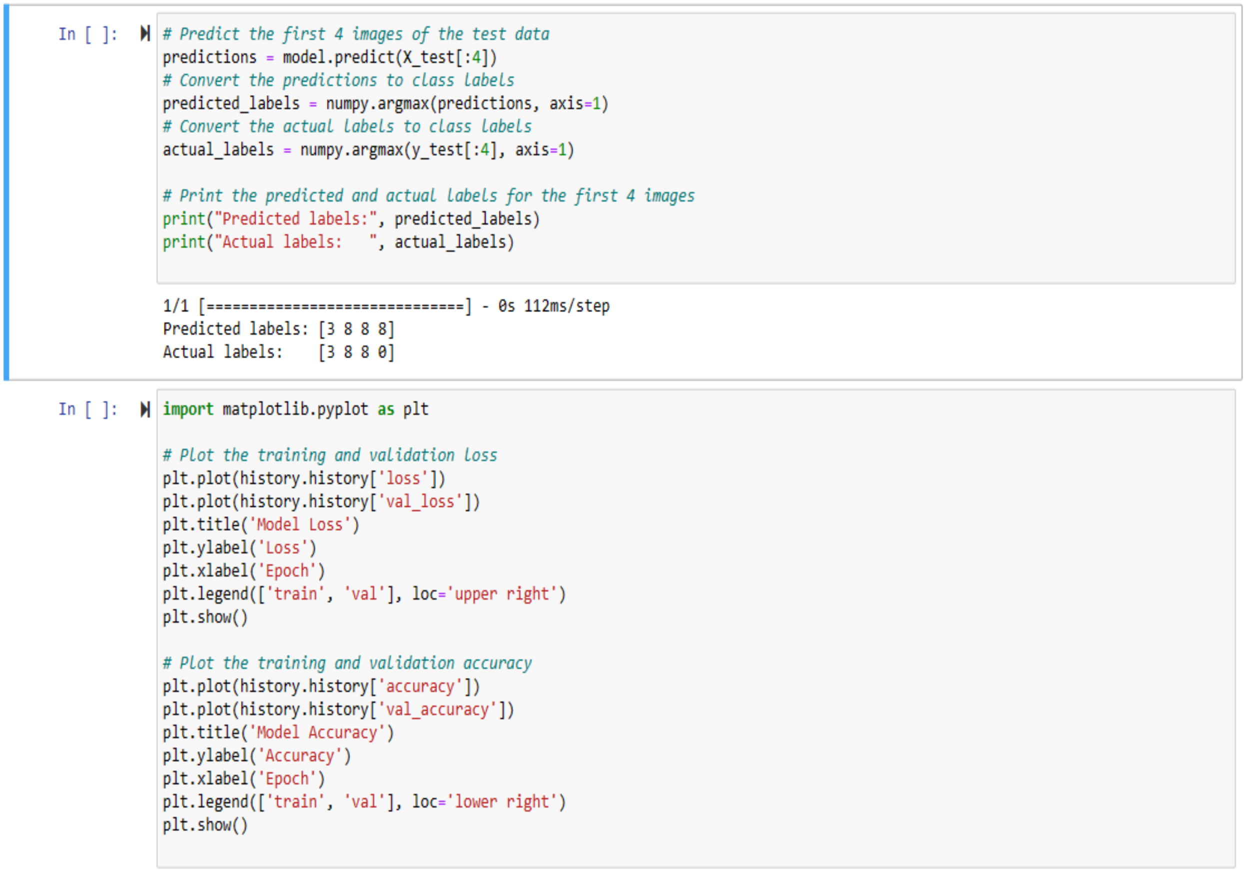
```
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 5
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate)
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))
```



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 32, 32, 32)        896

 dropout_8 (Dropout)         (None, 32, 32, 32)        0

 conv2d_9 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d_4 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 conv2d_10 (Conv2D)          (None, 16, 16, 64)        18496

 dropout_9 (Dropout)         (None, 16, 16, 64)        0

 conv2d_11 (Conv2D)          (None, 16, 16, 64)        36928

 max_pooling2d_5 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_12 (Conv2D)          (None, 8, 8, 128)         73856

 dropout_10 (Dropout)        (None, 8, 8, 128)         0

 conv2d_13 (Conv2D)          (None, 8, 8, 128)         147584

 max_pooling2d_6 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten_2 (Flatten)         (None, 2048)              0

 dropout_11 (Dropout)        (None, 2048)              0

 dense_5 (Dense)             (None, 1024)              2098176

 dropout_12 (Dropout)        (None, 1024)              0

 dense_6 (Dense)             (None, 512)               524800

 dropout_13 (Dropout)        (None, 512)               0

 dense_7 (Dense)             (None, 10)                5130
```



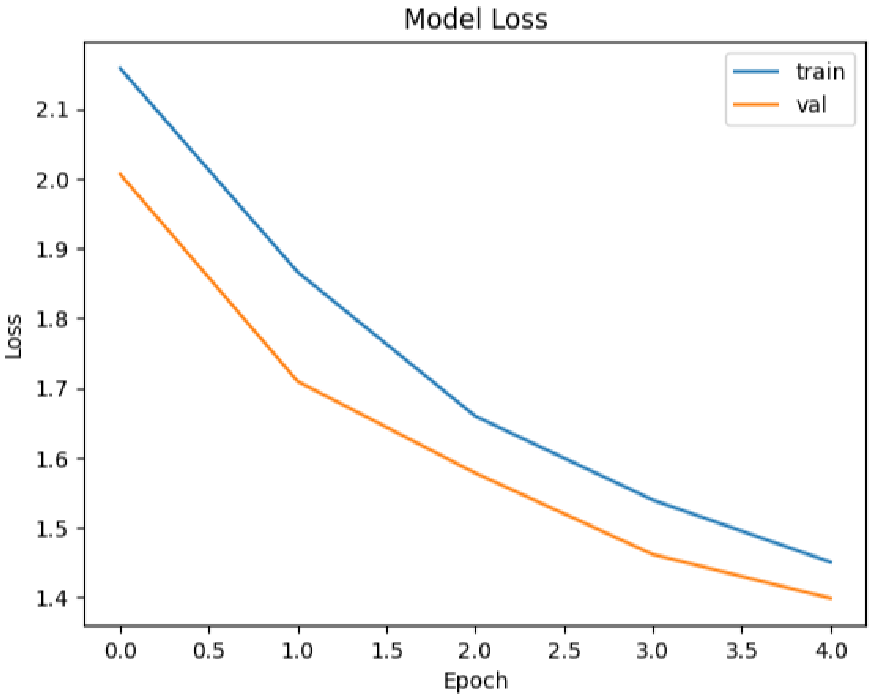
```
=================================================================
Total params: 2915114 (11.12 MB)
Trainable params: 2915114 (11.12 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
Epoch 1/5
1563/1563 [==============================] - 14s 8ms/step - loss: 2.1582 - accuracy: 0.1918 - val_loss: 2.0066 - val_accurac
y: 0.2972
Epoch 2/5
1563/1563 [==============================] - 12s 8ms/step - loss: 1.8660 - accuracy: 0.3234 - val_loss: 1.7094 - val_accurac
y: 0.3782
Epoch 3/5
1563/1563 [==============================] - 12s 8ms/step - loss: 1.6595 - accuracy: 0.3948 - val_loss: 1.5780 - val_accurac
y: 0.4422
Epoch 4/5
1563/1563 [==============================] - 12s 7ms/step - loss: 1.5394 - accuracy: 0.4419 - val_loss: 1.4614 - val_accurac
y: 0.4809
Epoch 5/5
1563/1563 [==============================] - 11s 7ms/step - loss: 1.4504 - accuracy: 0.4737 - val_loss: 1.3986 - val_accurac
y: 0.5071
Accuracy: 50.71%
```

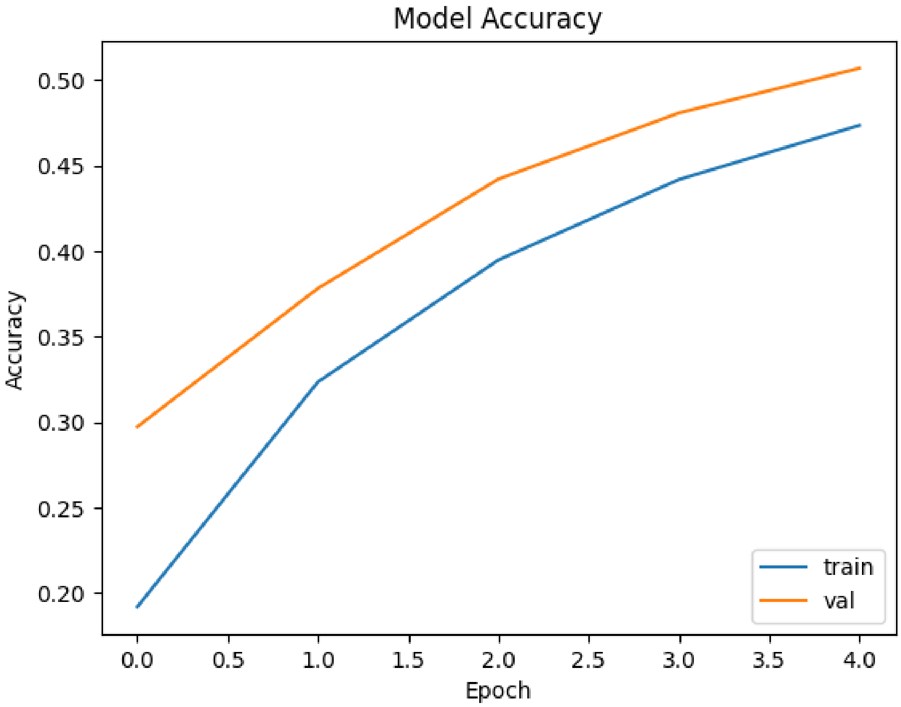


```python
# Predict the first 4 images of the test data
predictions = model.predict(X_test[:4])
# Convert the predictions to class labels
predicted_labels = numpy.argmax(predictions, axis=1)
# Convert the actual labels to class labels
actual_labels = numpy.argmax(y_test[:4], axis=1)

# Print the predicted and actual labels for the first 4 images
print("Predicted labels:", predicted_labels)
print("Actual labels:   ", actual_labels)
```

```
1/1 [==============================] - 0s 112ms/step
Predicted labels: [3 8 8 8]
Actual labels:    [3 8 8 0]
```

```python
import matplotlib.pyplot as plt

# Plot the training and validation loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()

# Plot the training and validation accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='lower right')
plt.show()
```




Model Accuracy
Accuracy
0.20
0.25
0.30
0.35
0.40
0.45
0.50
Epoch
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
train
val