Name: Vaarshinni Reddy Andru

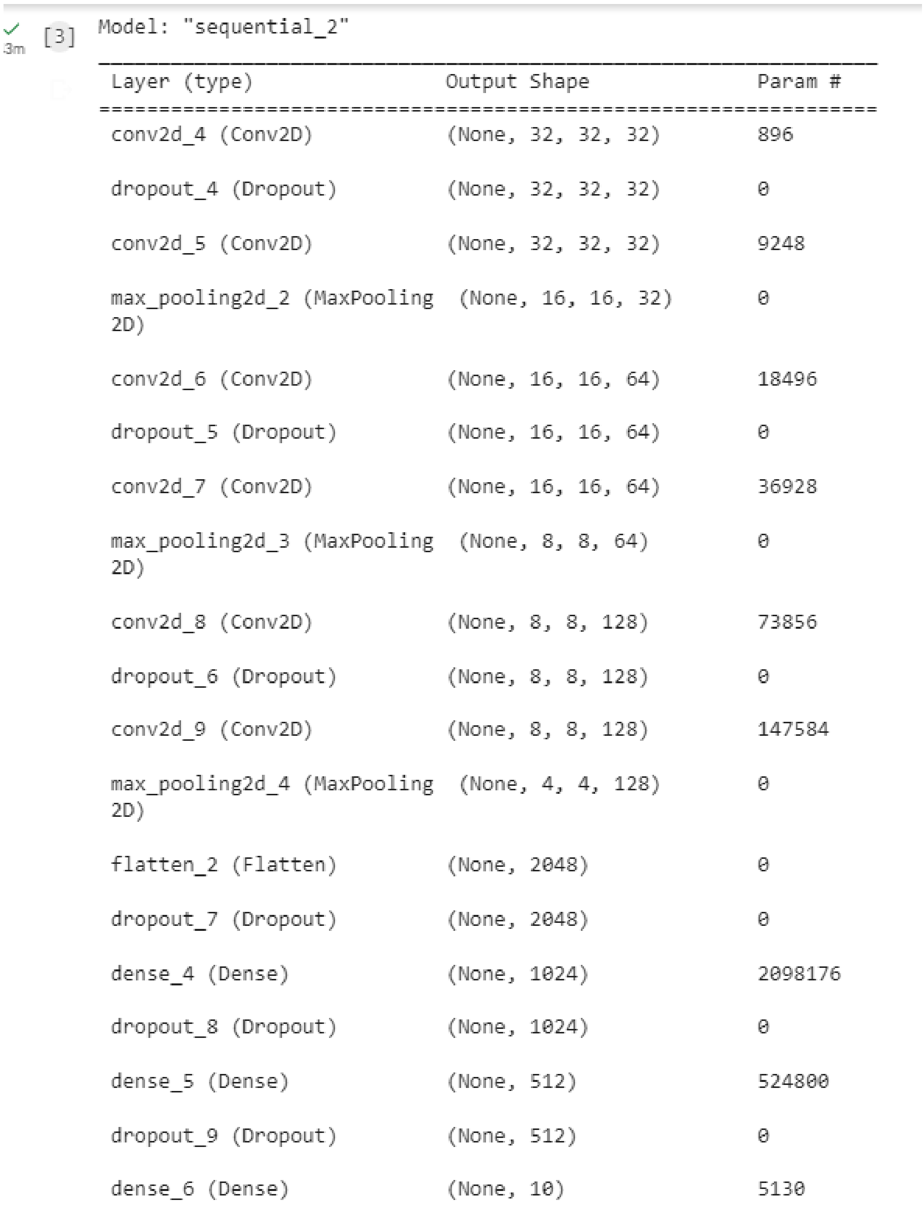
Enroll no: 700742952

Follow the instruction below and then report how the performance changed. (apply all at once)

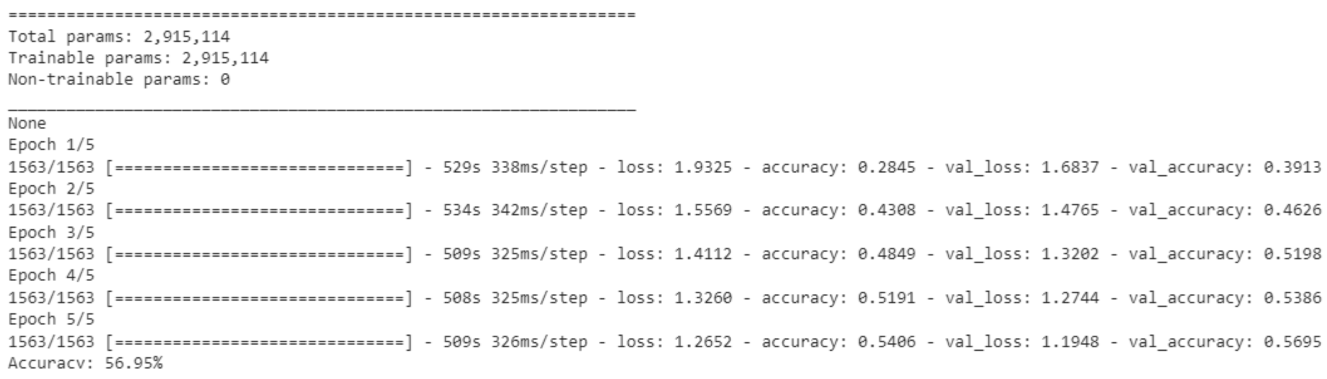
- Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
- Max Pool layer with size 2×2.
- Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
- Max Pool layer with size 2×2.
- Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer,128 feature maps with a size of 3×3 and a rectifier activation function.
- Max Pool layer with size 2×2.
- Flatten layer.
- Dropout layer at 20%.
- Fully connected layer with 1024 units and a rectifier activation function.

- Dropout layer at 20%.
- Fully connected layer with 512 units and a rectifier activation function.
- Dropout layer at 20%.
- Fully connected output layer with 10 units and a Softmax activation function

Did the performance change?

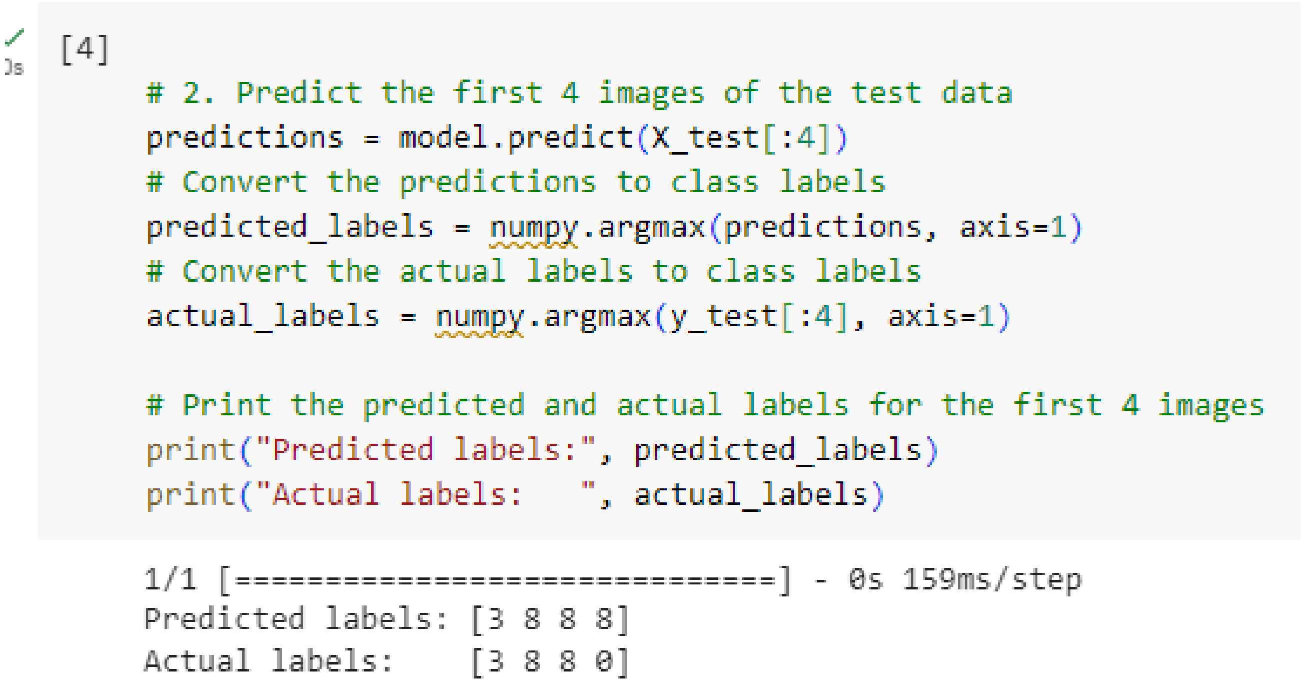


```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 32, 32, 32)        896

 dropout_4 (Dropout)         (None, 32, 32, 32)        0

 conv2d_5 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d_2 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 conv2d_6 (Conv2D)           (None, 16, 16, 64)        18496

 dropout_5 (Dropout)         (None, 16, 16, 64)        0

 conv2d_7 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 conv2d_8 (Conv2D)           (None, 8, 8, 128)         73856

 dropout_6 (Dropout)         (None, 8, 8, 128)         0

 conv2d_9 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_4 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten_2 (Flatten)         (None, 2048)              0

 dropout_7 (Dropout)         (None, 2048)              0

 dense_4 (Dense)             (None, 1024)              2098176

 dropout_8 (Dropout)         (None, 1024)              0

 dense_5 (Dense)             (None, 512)               524800

 dropout_9 (Dropout)         (None, 512)               0

 dense_6 (Dense)             (None, 10)                5130
```



```
=================================================================
Total params: 2,915,114
Trainable params: 2,915,114
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/5
1563/1563 [==============================] - 529s 338ms/step - loss: 1.9325 - accuracy: 0.2845 - val_loss: 1.6837 - val_accuracy: 0.3913
Epoch 2/5
1563/1563 [==============================] - 534s 342ms/step - loss: 1.5569 - accuracy: 0.4308 - val_loss: 1.4765 - val_accuracy: 0.4626
Epoch 3/5
1563/1563 [==============================] - 509s 325ms/step - loss: 1.4112 - accuracy: 0.4849 - val_loss: 1.3202 - val_accuracy: 0.5198
Epoch 4/5
1563/1563 [==============================] - 508s 325ms/step - loss: 1.3260 - accuracy: 0.5191 - val_loss: 1.2744 - val_accuracy: 0.5386
Epoch 5/5
1563/1563 [==============================] - 509s 326ms/step - loss: 1.2652 - accuracy: 0.5406 - val_loss: 1.1948 - val_accuracy: 0.5695
Accuracy: 56.95%
```

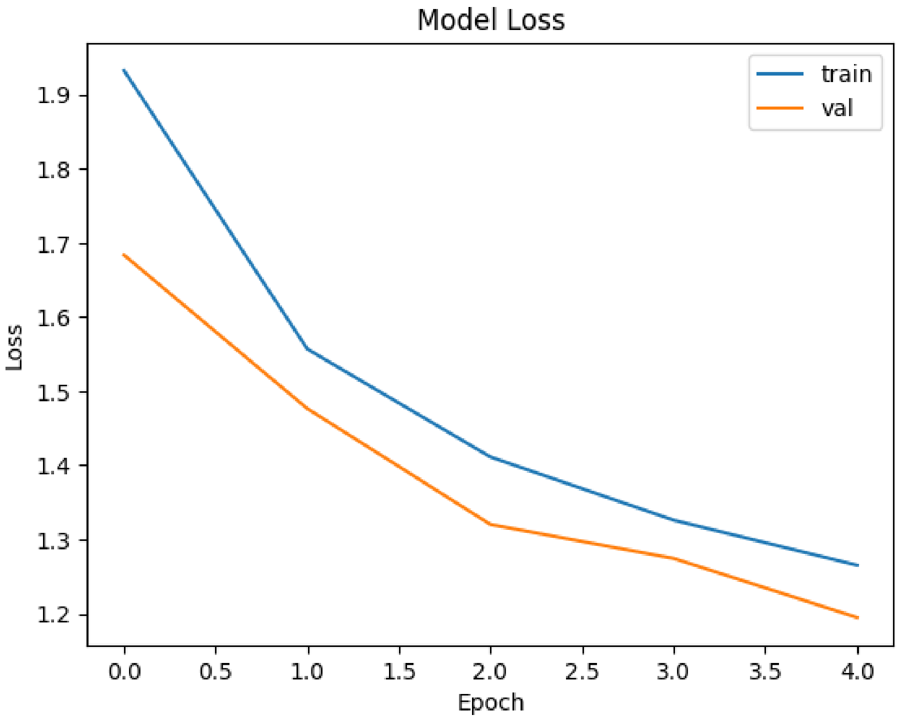
2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly.

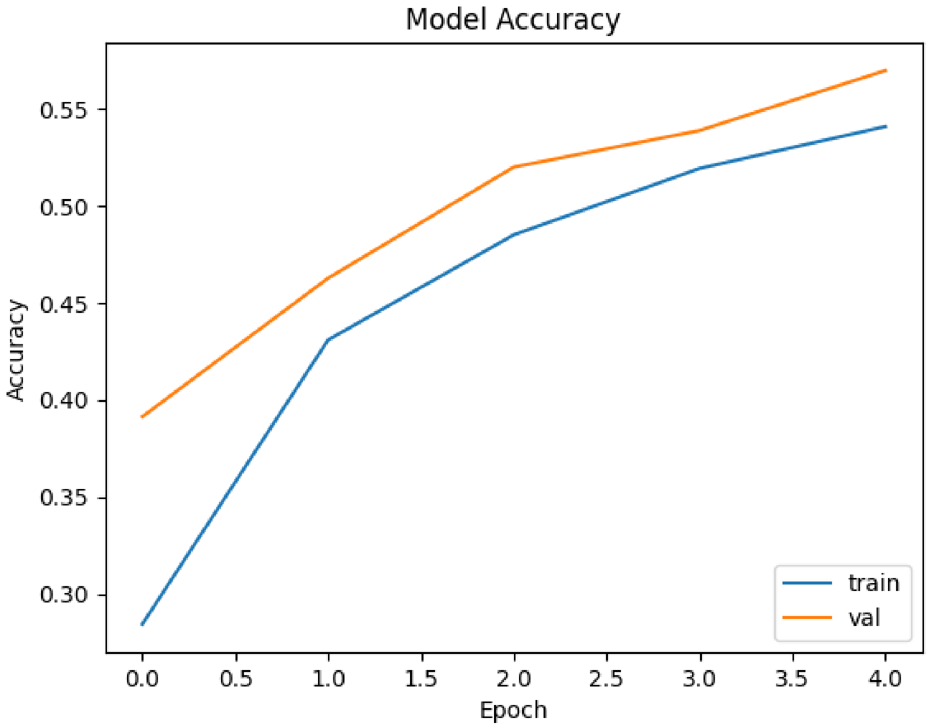


```
# 2. Predict the first 4 images of the test data
predictions = model.predict(X_test[:4])
# Convert the predictions to class labels
predicted_labels = numpy.argmax(predictions, axis=1)
# Convert the actual labels to class labels
actual_labels = numpy.argmax(y_test[:4], axis=1)

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

```
1/1 [==============================] - 0s 159ms/step
Predicted labels: [3 8 8 8]
Actual labels:    [3 8 8 0]
```

3. Visualize Loss and Accuracy using the history object


Model Loss
train
val
Loss
Epoch


Model Accuracy
Accuracy
train
val
Epoch