Name: Dheeraj Kodakandla

Network id: 700738796

Video link:

https://drive.google.com/file/d/1K5JWpKwS4JIfUjHilQ2-hRUePk8QFADJ/view?usp=sharing

In the previous code provided the code has wrong values (3,32,32), so I have changed to (32,32,3) then the code has executed successfully.

```
170498071/170498071 [==============================] - 2s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 dropout (Dropout)           (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 512)               4194816

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 4,210,090
Trainable params: 4,210,090
Non-trainable params: 0
_________________________________________________________________
None
```

```
epochs = 5
batch_size = 32
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=batch_size)
```

```
Epoch 1/5
1563/1563 [==============================] - 21s 7ms/step - loss: 1.7416 - accuracy: 0.3685 - val_loss: 1.4289 - val_accuracy: 0.4907
Epoch 2/5
1563/1563 [==============================] - 10s 7ms/step - loss: 1.3969 - accuracy: 0.4991 - val_loss: 1.2436 - val_accuracy: 0.5622
Epoch 3/5
1563/1563 [==============================] - 10s 6ms/step - loss: 1.2315 - accuracy: 0.5614 - val_loss: 1.1621 - val_accuracy: 0.5867
Epoch 4/5
1563/1563 [==============================] - 11s 7ms/step - loss: 1.0986 - accuracy: 0.6107 - val_loss: 1.0556 - val_accuracy: 0.6256
Epoch 5/5
1563/1563 [==============================] - 10s 6ms/step - loss: 0.9814 - accuracy: 0.6541 - val_loss: 0.9858 - val_accuracy: 0.6522

<keras.callbacks.History at 0x7f9e30281100>
```

```
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Python

```
Accuracy: 65.22%
```

1. 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.

Here we have compiled the model after changing the code with the above changes and fit the model and executed it.

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

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

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

 dropout_10 (Dropout)        (None, 8, 8, 128)         0
...
1563/1563 [==============================] - 13s 8ms/step - loss: 1.2989 - accuracy: 0.5308 - val_loss: 1.2250 - val_accuracy: 0.5626
Epoch 5/5
1563/1563 [==============================] - 12s 8ms/step - loss: 1.2454 - accuracy: 0.5513 - val_loss: 1.1979 - val_accuracy: 0.5646
Accuracy: 56.46%
```

Did the performance change?

Here we can see the accuracy has changed from 65.22% to 56.46%. performance has changed.

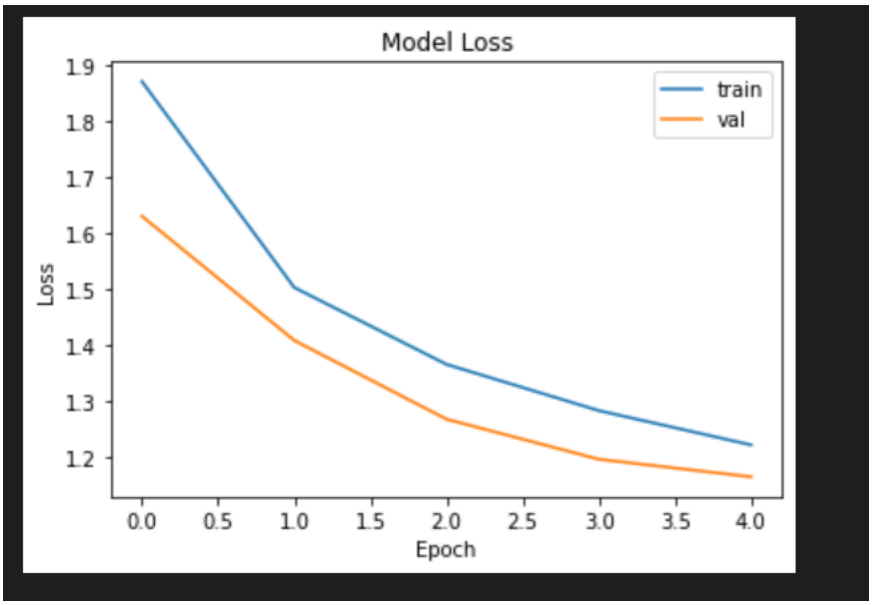
2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly

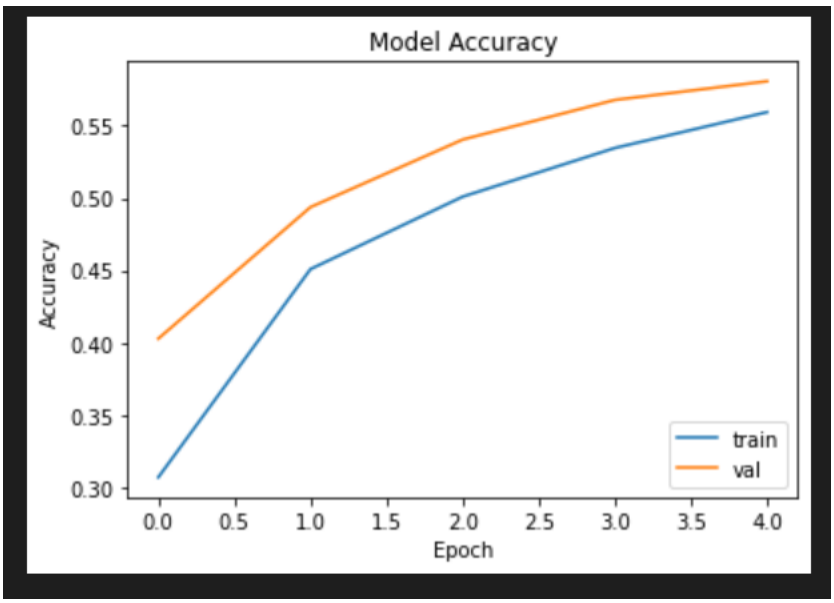
First we have predicted the first 4 images of the test data and then converted the predictions to class labels and then converted the actual labels to class labels and then printed the predicted and actual labels for the first four images,

```
1/1 [==============================] - 0s 270ms/step
Predicted labels: [5 8 8 8]
Actual labels:    [3 8 8 0]
```

3. Visualize Loss and Accuracy using the history object

Here we have plotted the model loss and model accuracy using the above code.


Model Loss
train
val
Loss
Epoch
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Model Accuracy
train
val
Accuracy
Epoch
0.55
0.50
0.45
0.40
0.35
0.30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0