MODEL DESIGN

CIFAR-10 is normally an UNDERFITTING MODEL. We worked on it from scratch to improve the model and achieved a training accuracy of 93% and testing accuracy of 90%.

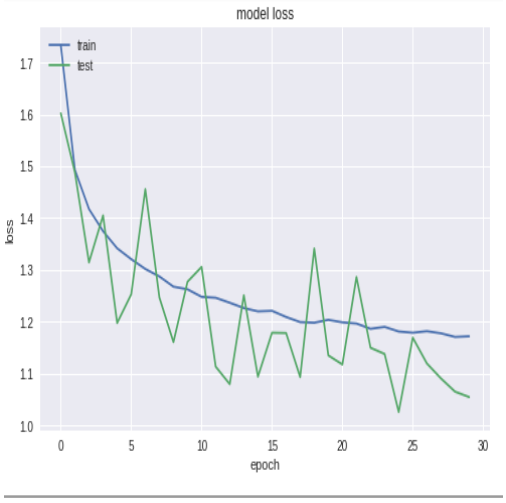
MODEL 1 :

Started from a simple 3 layer Covnets architecture.
Model specification :
3 Convolutional layer : 32,64,128
Filters : 3x3
Dropout : 0.5

```
model.add(Conv2D(32,(3,3),activation = 'relu' ,input_shape=[32,32,3]))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.5))
model.add(Conv2D(64,(3,3),activation = 'relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.5))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(512))
model.add(Dropout(0.2))
model.add(Dense(10,activation='Softmax'))
```

Loss : Categorical_crossentropy
Optimizer : Adam

This model training for 50 epochs and batch size of 32 gave training accuracy of 58% and testing of 63% - UNDERFITTING Model.



MODEL 2 :

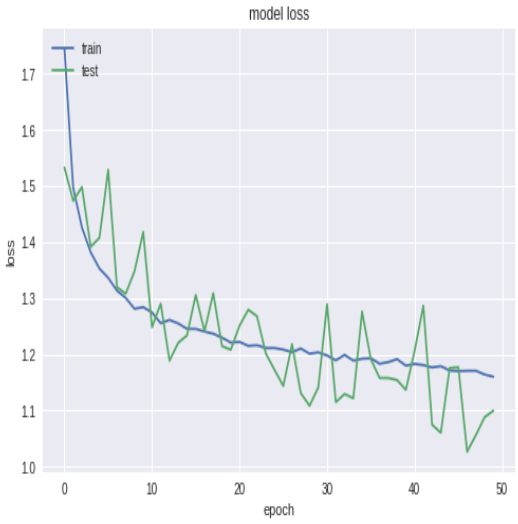
To overcome the Underfitting problem :

1. More data can be added – Data Augmentation
2. More layers can be added

In this model, in order to increase the data I have used Data Augmentation keeping the architecture same.

```
datagen = ImageDataGenerator(
featurewise_center=True,
featurewise_std_normalization=True,
rotation_range=20,
width_shift_range=0.2,
height_shift_range=0.2,
horizontal_flip=True,
vertical_flip = False)
```

For batch size of 32 and epochs = 50, this model gave a training accuracy of 49% and testing accuracy of 11% which is now Overfitting.



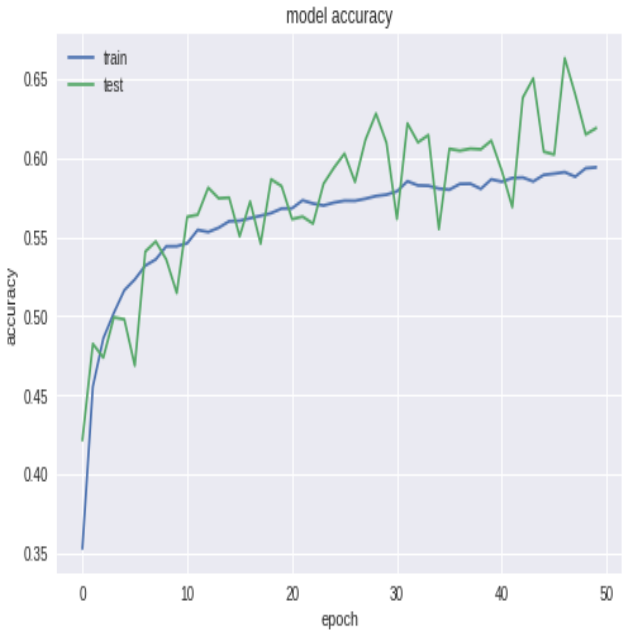
MODEL 3 :

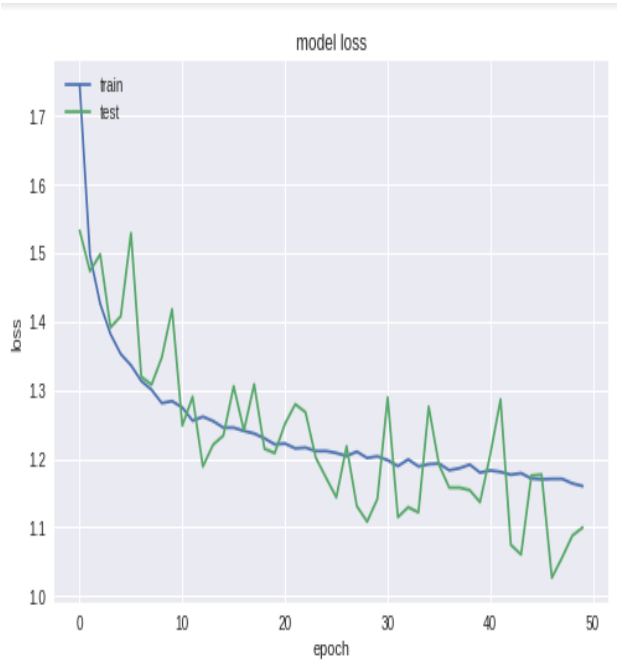
To overcome this overfitting :

1. BatchNormalization
2. Regularizers
3. Dropout layers

In this model, I have added Regularizers.

There are two types of Regularizers, L1 and L2.
First I have tried with L1 regularizer.
This model with batch size = 32 and number of epochs = 50 gave an over-fitting model.
Training accuracy – 25%
Testing accuracy – 15%



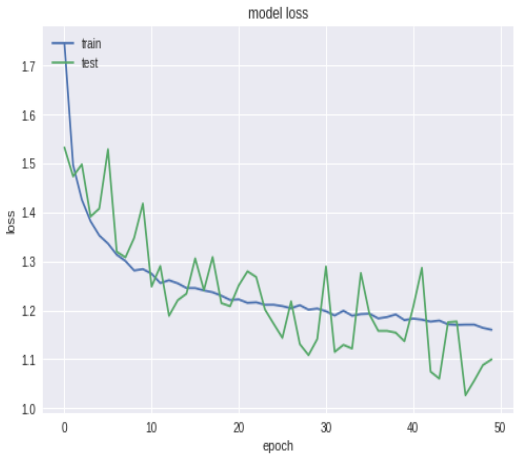


Still no improvement and the training accuracy is not improving after 30th epoch.

MODEL 4 :

Tried increasing the number of convolutional layers.
Added additional Conv2D(32),Conv2D(64) and Conv2D(128) layer with data augmentation.

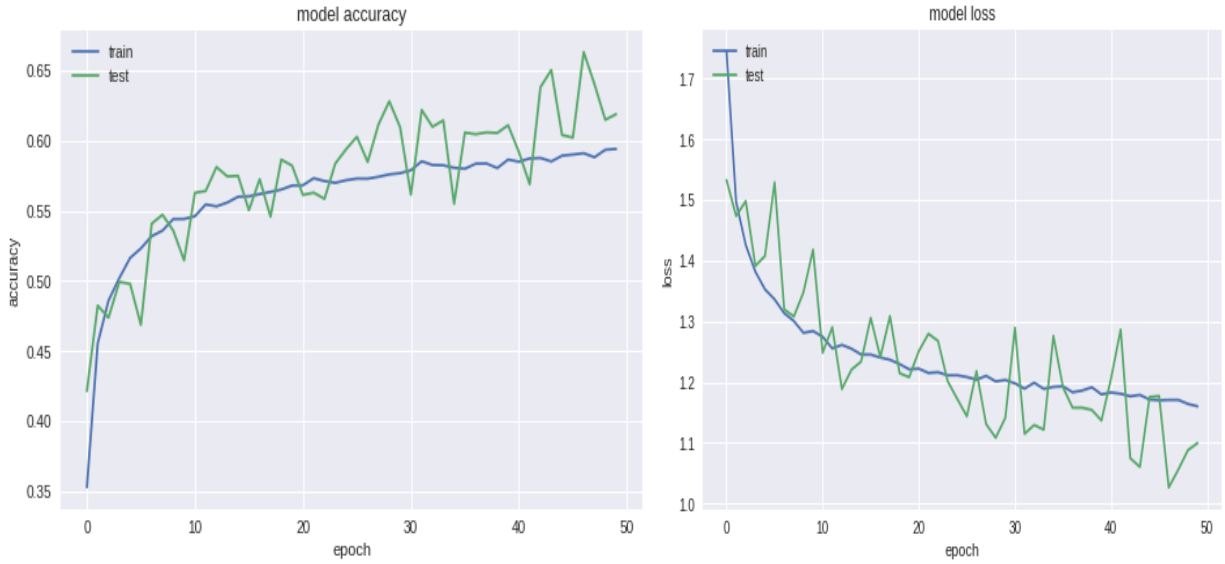
This gave an ideal model with poor accuracy score.
Training accuracy – 9%
Testing accuracy – 10%



From the Accuracy-epochs graph, we can say that the training accuracy hasn't improved much after 30th epoch which was the case in the last model too.

MODEL 5 :

In order to overcome that, in this model L2 regularizer is used with weight decay= 0.01.
This model gave a training accuracy of – 38% and testing accuracy of 10% which is an overfitting model.

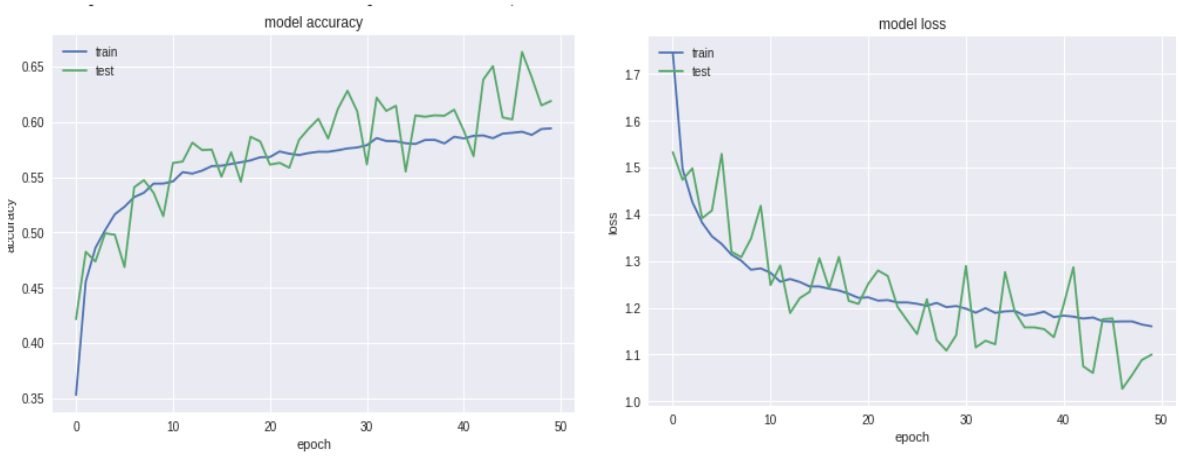


Even in this model, the accuracy is not improving after a certain number of epochs and test accuracy is fluctuating.

MODEL 6 :

Previous model gave a poor accuracy scores and also fluctuating test accuracy. In order to improve on that model, In this model weight decay is increased to 0.1. By changing weight decay, fluctuating might improve.

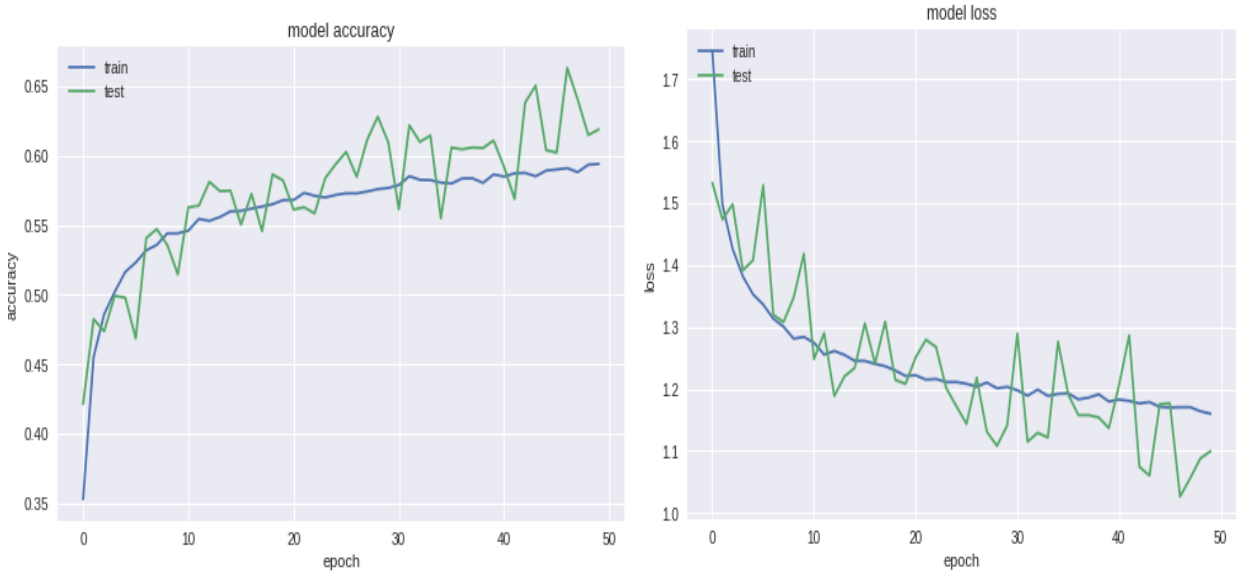
This model gave a training accuracy – 38% and test accuracy – 10% which is similar to the previous model and the test accuracy is still fluctuating.



MODEL 7 :

Model 6 didn't give any different result from model 5. So, in this model the weight decay rate is reduced to 0.001 and changing featurewise_center and featurewise_std_normalization = False.

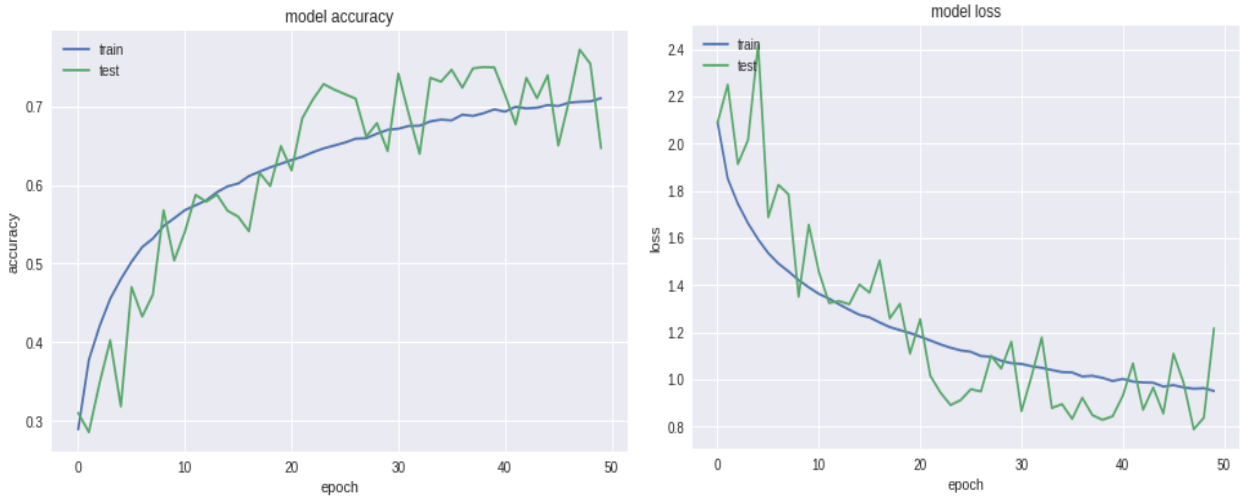
For batch size of 32 and epochs as 50, training accuracy is 72% and test accuracy is 24%.
This model gave a slightly better result than the other models but still overfitting.



MODEL 8 :

To improve on over-fitting, Optimizer is changed to SGD(learning rate = 0.01,decay = 1e-6,moemtum=0.9) from Adam and weight decay = 0.0005.

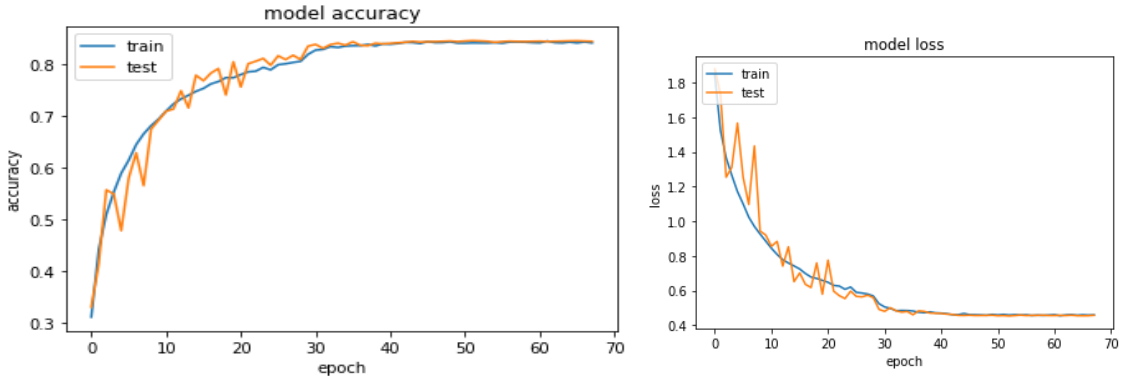
This model with batch size = 32 and number of epochs = 50 gave training accuracy of 71% and test accuracy of 64%.



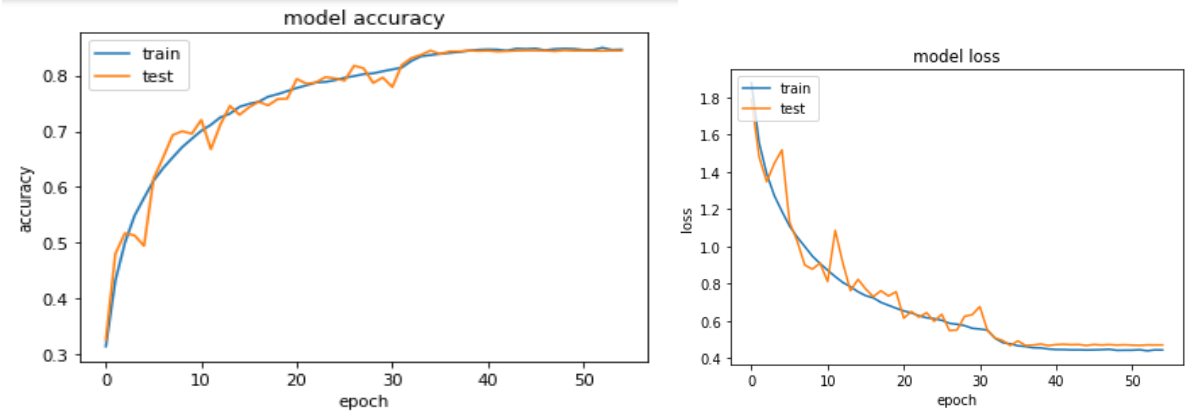
Though this model is giving a good accuracy, it is still overfitting.

MODEL 9:
2 more layers of Conv2D(64) are added to compromise on Overfitting.

This model gave a training acuracy of 84% and test accuracy of 84%.
This is an ideal model.



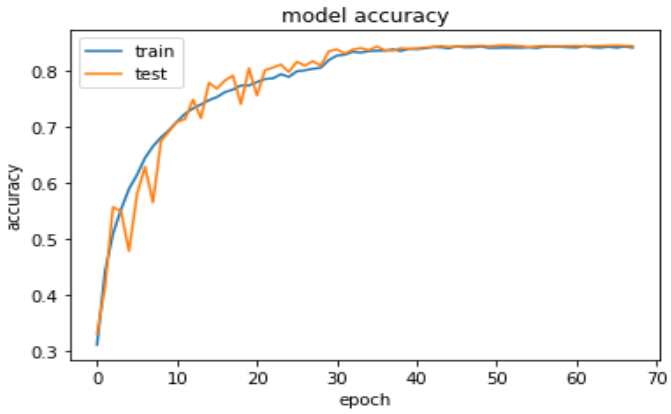
MODEL 10 :
Keeping the same model as experiment 9, chaniging the learning rate to 0.015 gave a trainign accuracy of 84% and test accuracy of 84%.

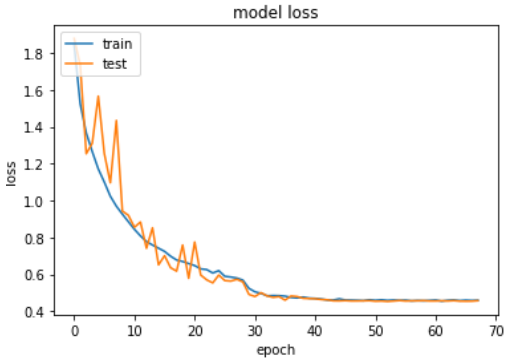


Though this experiment gave an ideal model, it didn't improve the accuracy.

MODEL 11 :
Adding extra 2 more layers of Conv2d(64) to check if the parameter increase has some effect on accuracy.

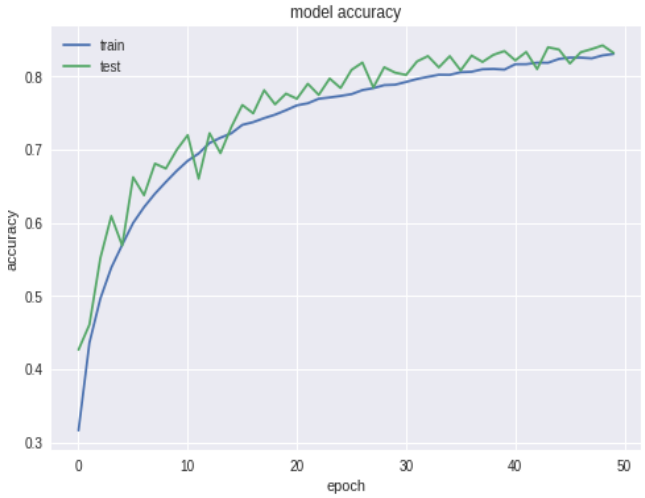
This model again gave us a training accuracy of 84% and test accuracy of 84%.
Though this is also an ideal mode, there is no improvement in the accuracy.

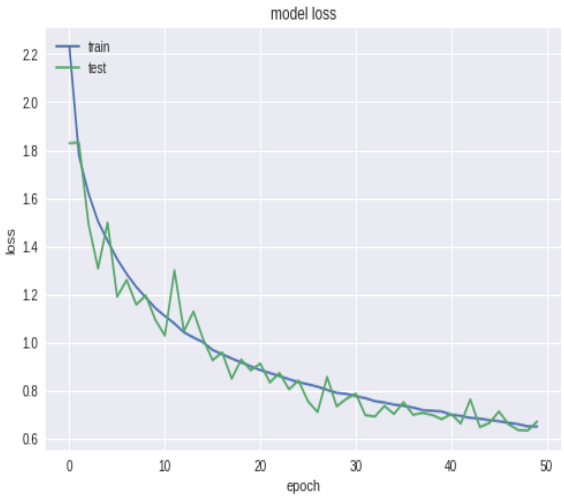




MODEL 9 :
To avoid overfitting, in this model 2 Conv2D(256) layers are added.

For batch size = 32 and number of epochs = 50, this model gave training accuracy of 83% and test accuracy of 83% which is an ideal model.

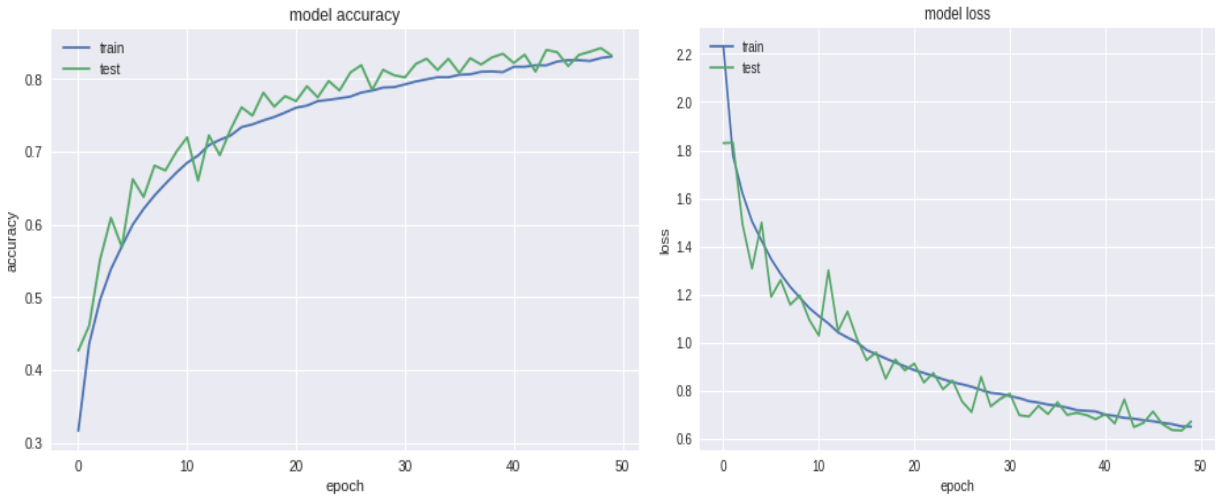




This can be considered as the base model and improvement can be made to this model to achieve a higher accuracy score.

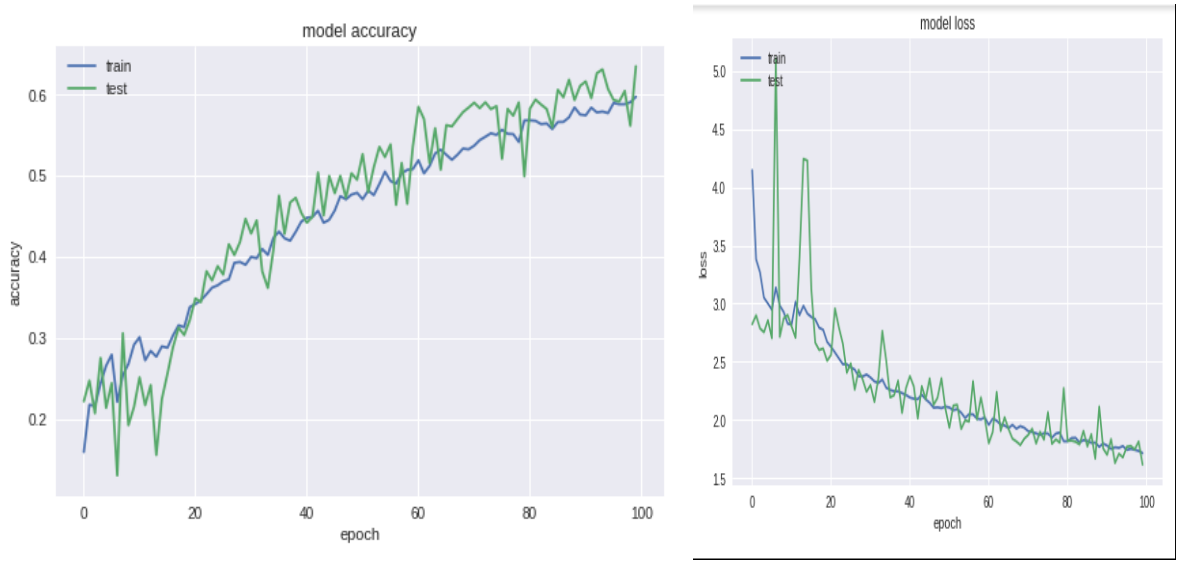
MODEL 10 :
As increasing number of Conv layers improved the accuracy, in this model 2 more layers of Conv2D(512) are added.

For batch size of 32 and epochs of 50, training accuracy is 82% and test accuracy is 84%.



MODEL 11 :

As increasing the number of Convolutional layer increases the accuracy, in this model an extra Conv2D(512) layer is added for batch size of 32 and number of epochs as 100.



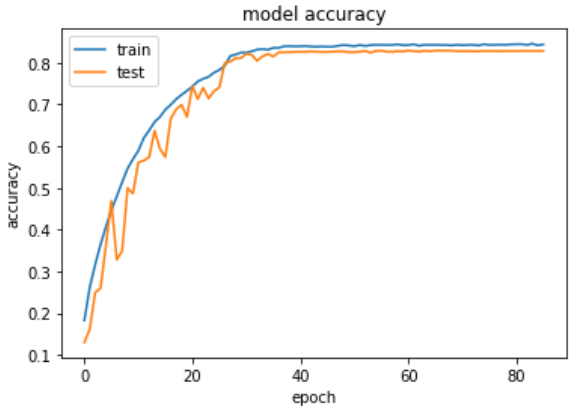
The model gave as training accuracy of 59% and test accuracy of 63%.

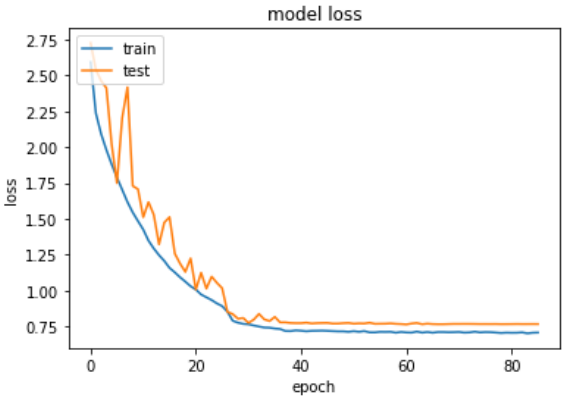
As, increasing the number of layers gave a bad result, no additional layers will be added.

MODEL 12 :

As increasing the parameters towards the end is giving better results, in this experiment number of parameters are increased from the first layer. That is first layer is Conv2D(64) then Conv2D(128), 4X Conv2D(256) and 6x Conv2D(512) and with callbacks (early stopping, reduceLRonPlateau)

This model gave us a training accuracy of 84% and test accuracy of 82% for number of epochs as 100 and batch size as 128.



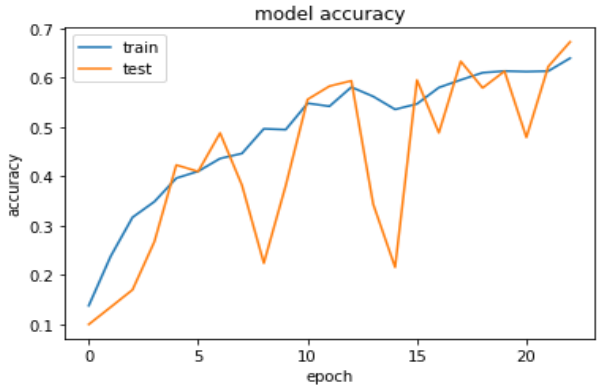


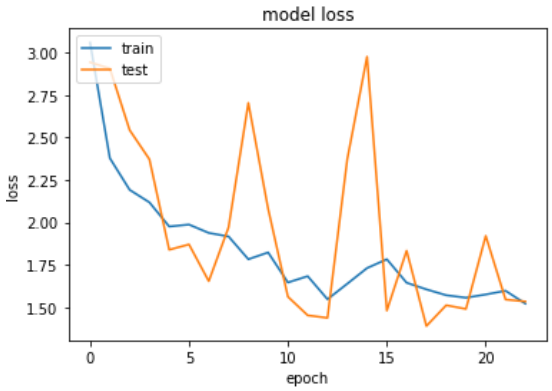
This model gave an accuracy less than other models.

MODEL 13 :

Same as model 12 but ran for 200 epochs.

It increased the training accuracy to 63% and test accuracy to 67%. This model is underfitting.

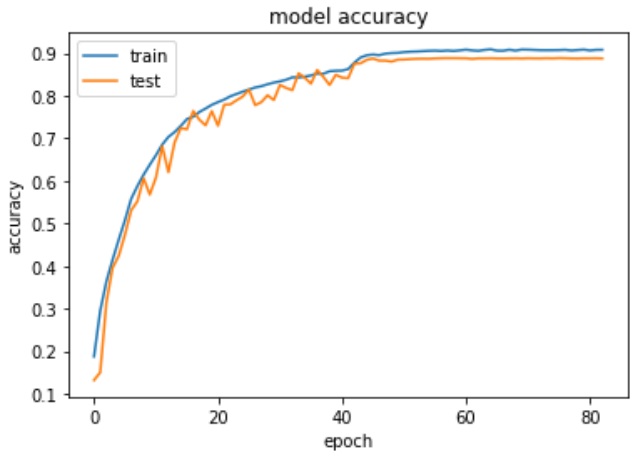


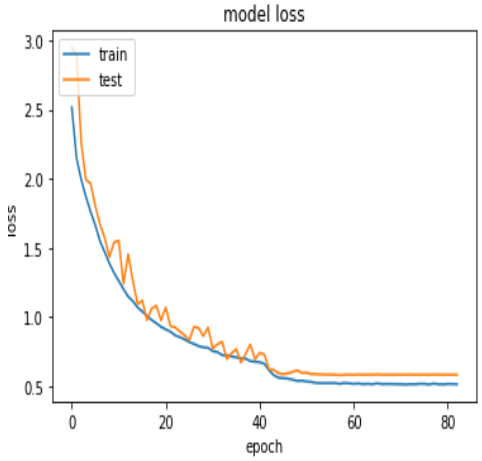


MODEL 14 :

Using the model same as 12 with dropout 0.25.

This model gave a testing accuracy of 90% and test accuracy of 88%. This is a very good model.



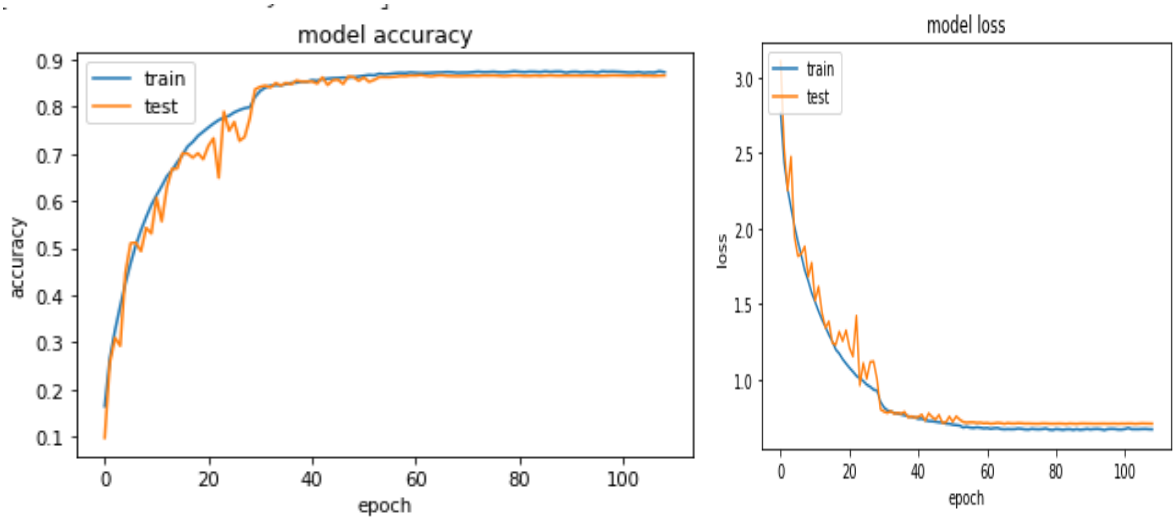


This is a very good model, but more experiments are conducted to check if a better accuracy can be achieved.

MODEL 15 :

Same as model 12, but with dropout of 0.3.

This model gives us a training accuracy of 87% and test accuracy of 86%.

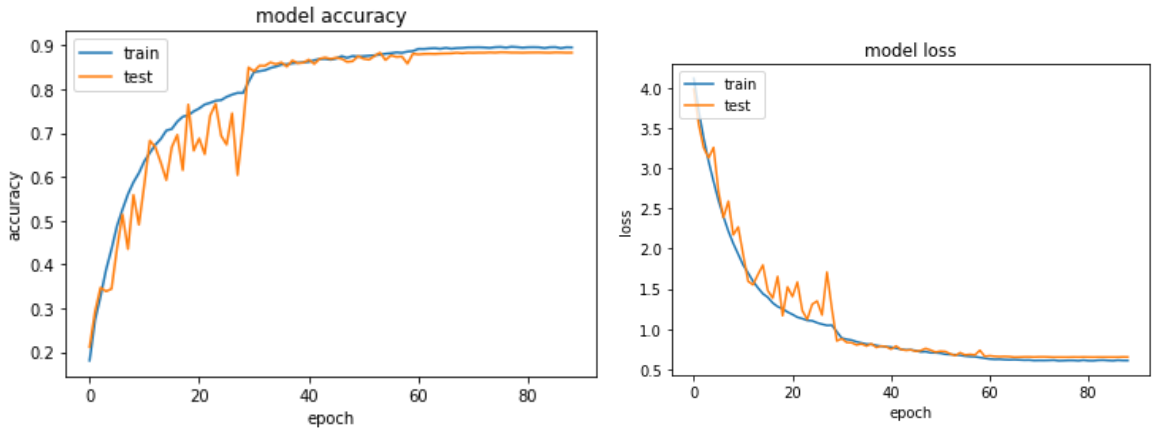


Though this model gives a good accuracy, it is lesser than the previous model.

MODEL 16 :

Same as model 12 but with dropout as 0.35.

This model gave training accuracy of 89% and test accuracy of 88%.

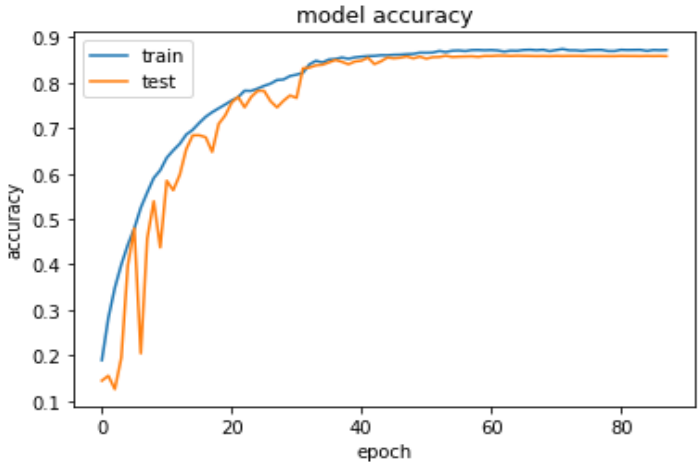


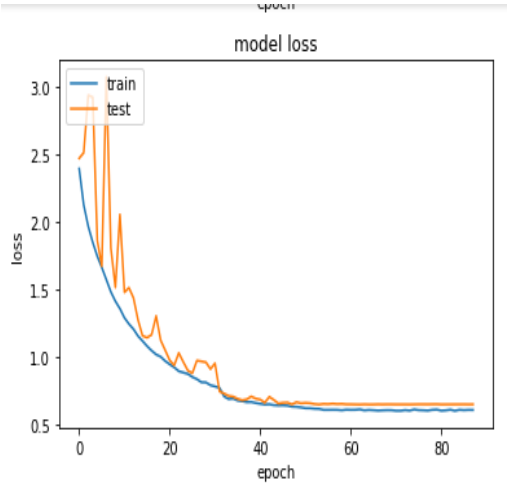
Good accuracy score but other experiments are conducted to check if this can be improved.

MODEL 17 :

Reducing the number of Conv2D(256) layers to 2 in model 12,

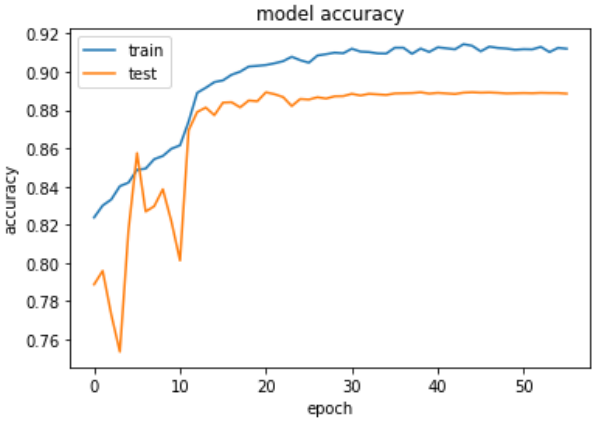
The training accuracy is 87% and test accuracy is 85%.

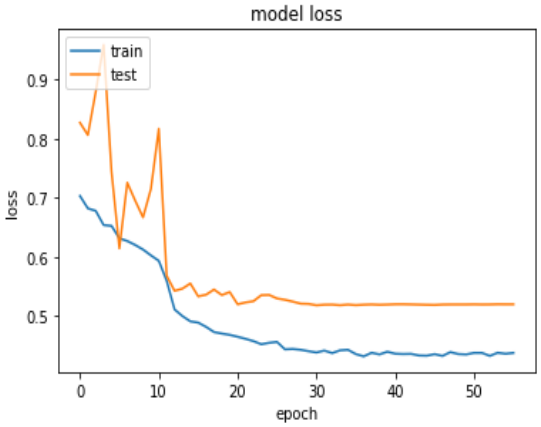




The model though has a good performance score, has accuracy lesser than other models.

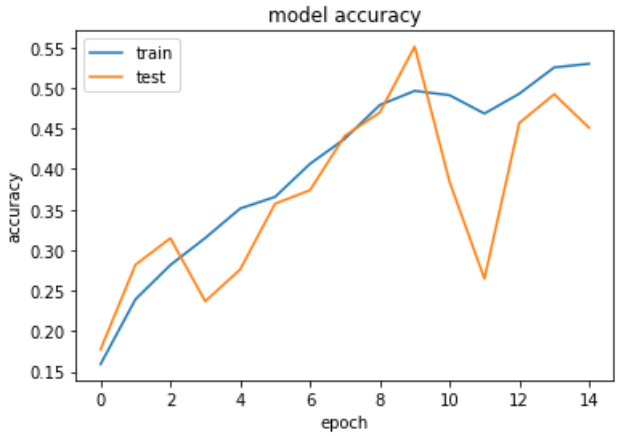
MODEL 18 :
Using the model in 12 but with learning rate as 0.015.
This model gave us a training accuracy of 91% and test accuracy of 88%.

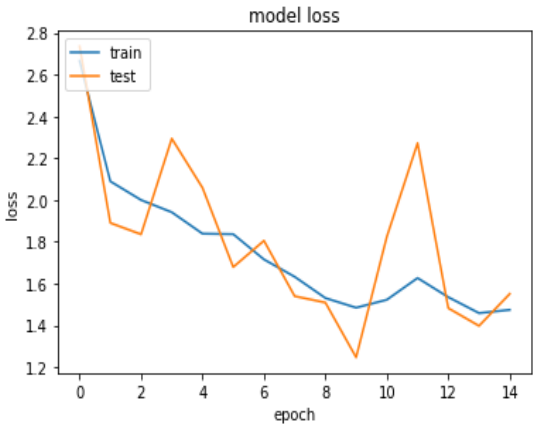




This model gave the best accuracy till now.

MODEL 19 :
In this model, the architecture is changes to first layer as 2xCon2D(64) and last layer as 2xConv2D(1024) with 2xConv2D(128), 2xConv2D(256) and 2xConv2D(512) in between them with callbacks same as model12.
This model gave us a training accuracy of 52% and test accuracy of 45%.





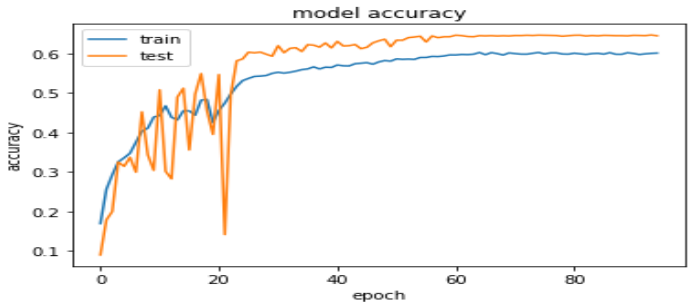
This model gave a poor accuracy.

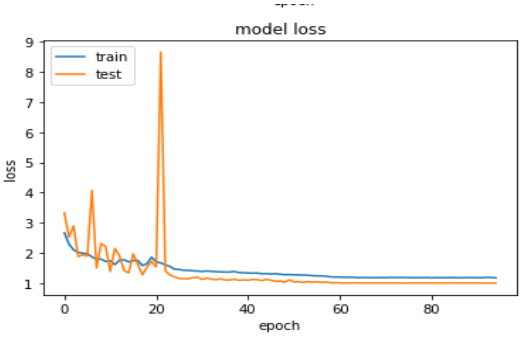
MODEL 20 :

In this model, architecture of 18 is used but the dropout values are changed to 0.3 in the hidden layers and 0.4 in the output layer which was earlier kept at 0.2.

This model gave a training accuracy of 60% and test accuracy of 64%.

Underfitting model.



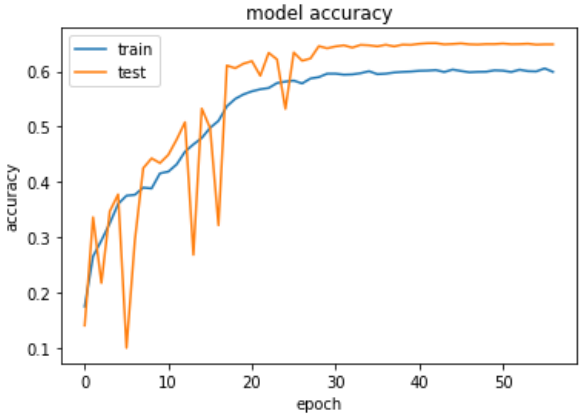


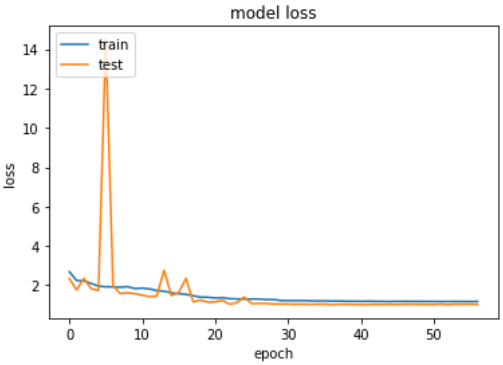
MODEL 21 :

In this experiment, the model is same as 18 but the dropout is changed to 0.1.

This model gave us a training accuracy of 59% and test accuracy of 64%.

Model has not performed well.

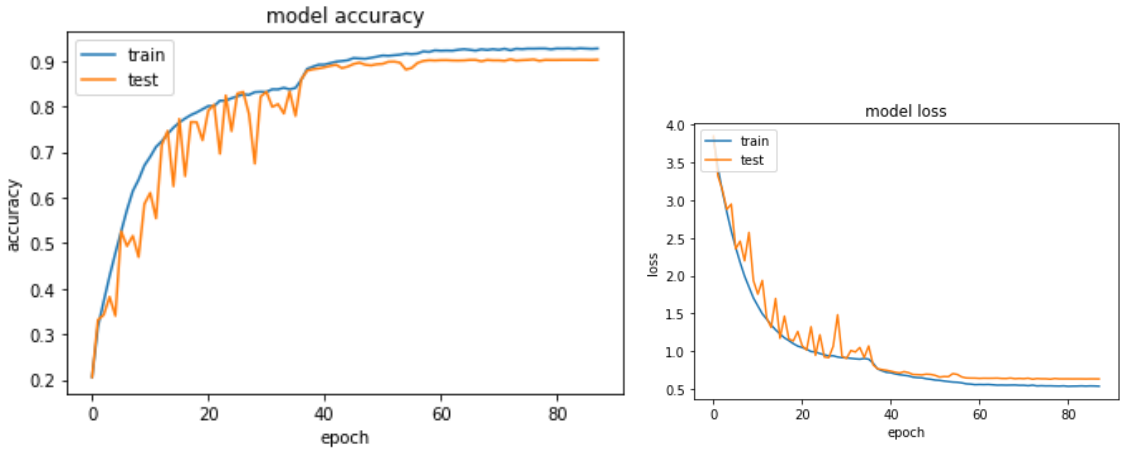




MODEL 22 :

Adding one extra layer of Conv2D(512) with weight decay as 0.001 and dropout as 0.35.

This model gave a training accuracy of 92% and test accuracy of 90%.

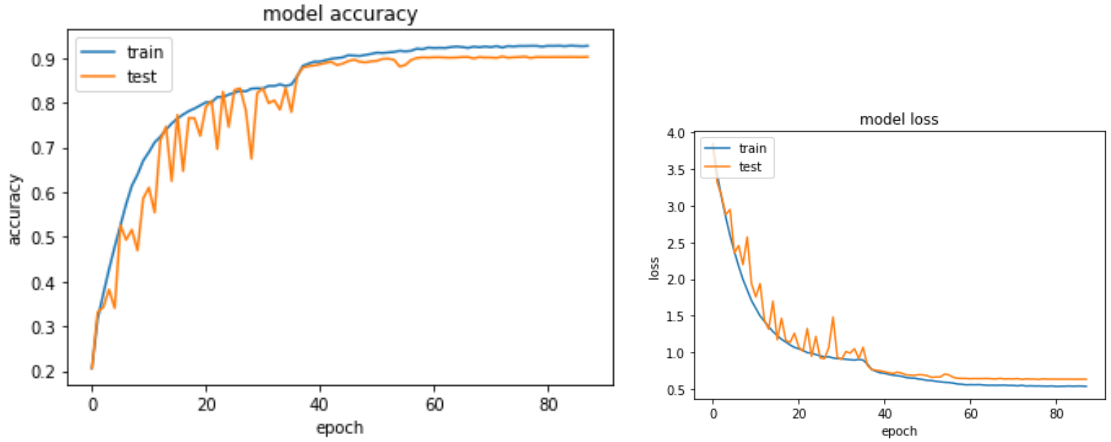


This model gave a very good accuracy. Still experimenting more to achieve a better performance.

MODEL 23 :

Keeping the architecture same as model 22 but with weight decay = 0.005 and dropout = 0.45

This model gave the same result as model 22. Training accuracy = 92% and test accuracy = 90%.



Increasing the dropout or weight decay doesn't have any impact on the accuracy after this point.

MODEL 24 :

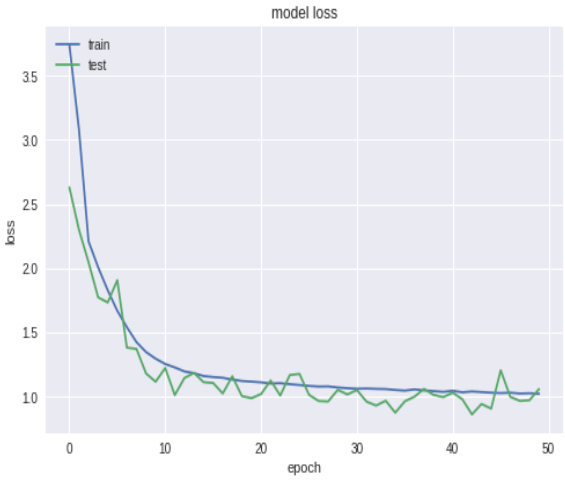
To check if Adam optimizer gives the same result or better result than SGD, this experiment was conducted.

Base model was the one used in Model 10, with epochs = 50.

With Adam optimizer :

1. Computation time increased. (Adam – 115 seconds ; SGD – 80 seconds)
2. Poor accuracy compared to SGD.

Training accuracy – 76% and Test accuracy is 75%.

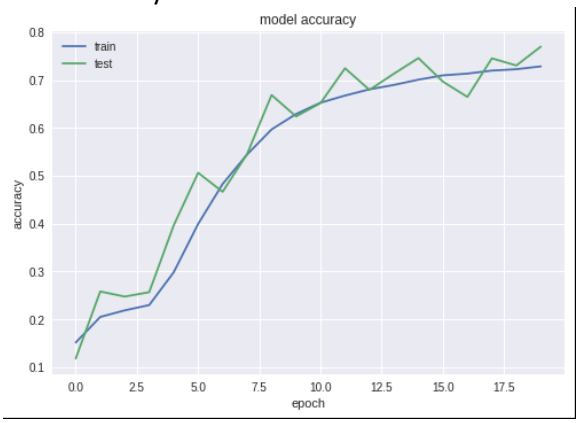


SGD will be the optimizer that will be used.

MODEL 25 :

Generally, BatchNormalization layer is added before introducing non-linearity into the model but few references has mentioned that using it after activation might improve the accuracy. So this experiment is conducted to check where to include the BatchNormalization layer.

Before activation layer :
Training accuracy – 71%
Test accuracy – 76%



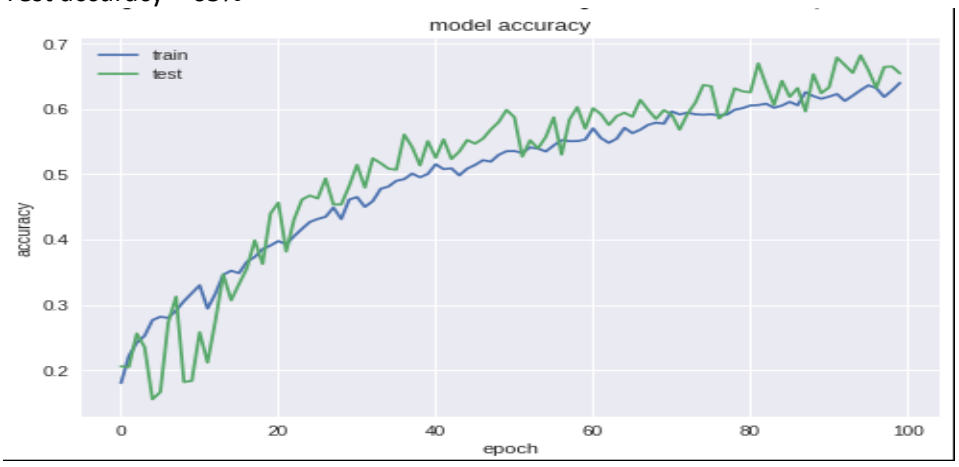
After activation layer :
Training accuracy – 73%
Test accuracy – 76%

Though both have a similar results, the latter is slightly better than the former one. So BatchNormalization layer will be included after introducing non-linearity in to the model.

MODEL 26 :
Experiments for deciding batch size and steps per epochs :

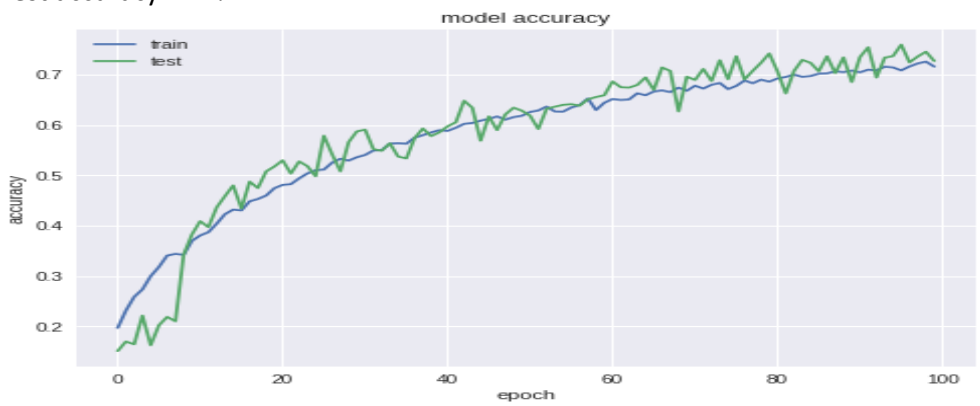
a) Steps per epochs and batch size = 32 :
   Same as model 10.

b) Steps per epochs and batch size = 256 :
   Training accuracy – 64%
   Test accuracy – 65%



c) Steps per epochs and batch size = 128 :
   Training accuracy – 80%
   Test accuracy – 80%

d) Steps per epochs = 128 and batch size = 32 :
Training accuracy – 72%
Test accuracy – 72%



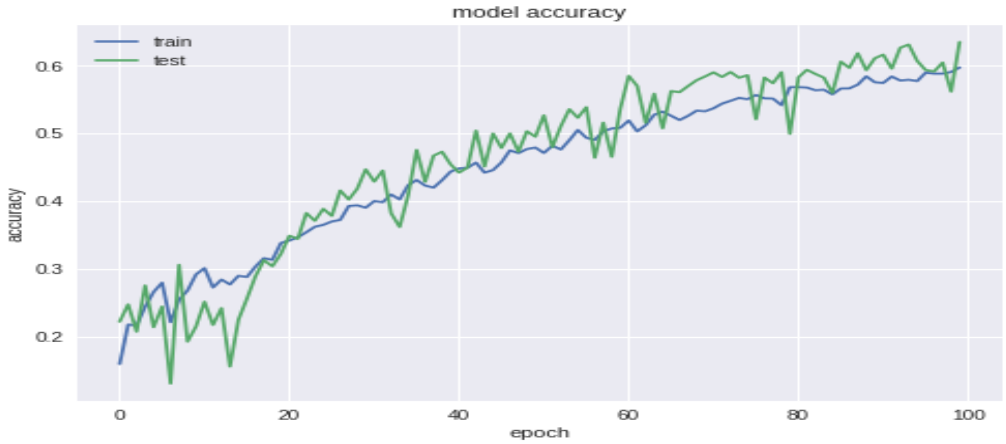
e) Steps per epochs = 128 and batch size = 64 :
Training accuracy – 79%
Test accuracy – 83%

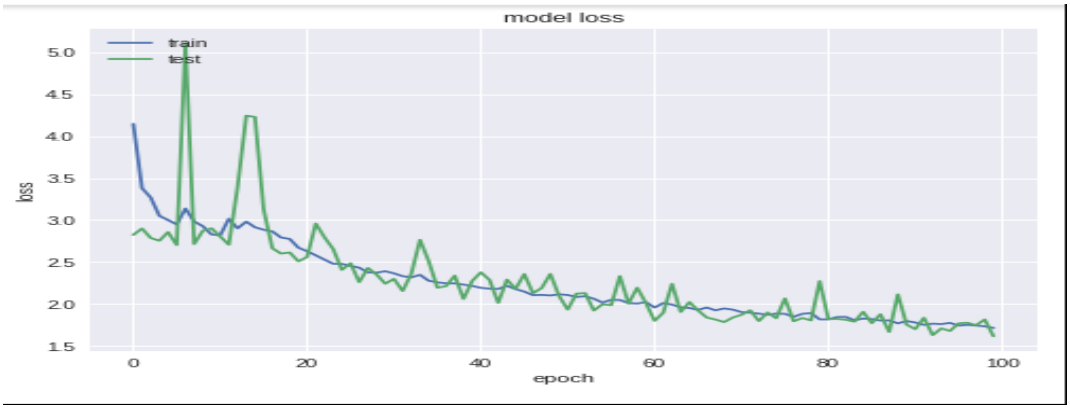
f) Steps per epoch = 32 and batch size = 64 :
Training accuracy – 86%
Test accuracy – 86%

Considering the above experiments, (f) has the best accuracy score.

MODEL 27 :
Fixing value for momentum :
Trying model with momentum = 1.0

This model gave a training accuracy – 59% and test accuracy – 63%.





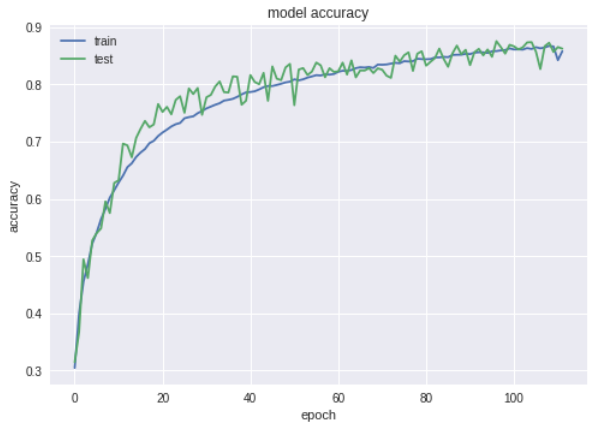
Momentum 0.9 will be a good number.

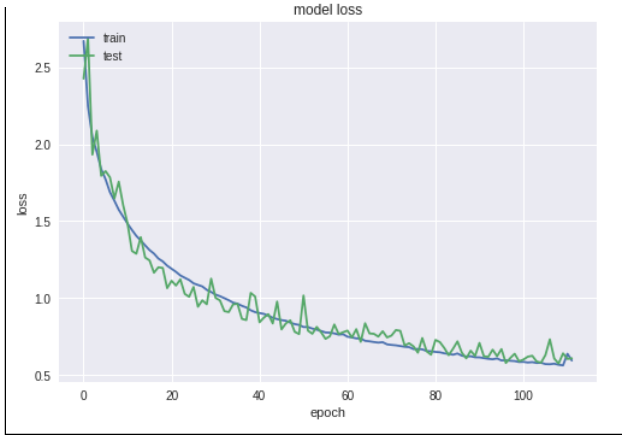
MODEL 28 :

Using Callbacks.

1. ReduceLRonPlateaue
2. Early Stopping

The model stopped at 112th epoch and gave a training accuracy of 85% and test accuracy of 86%.





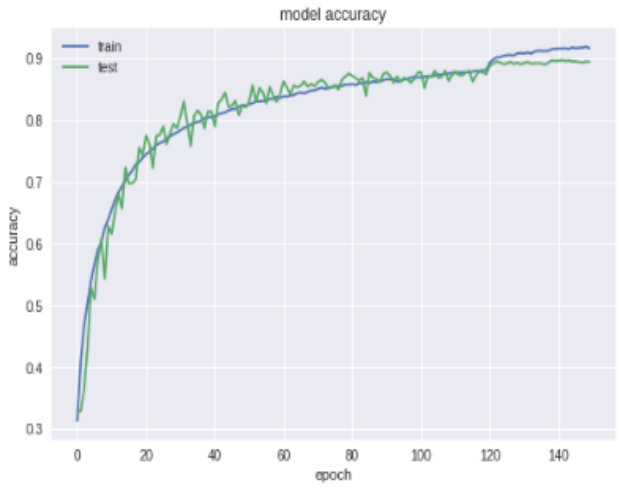

```
Epoch 103/150
1563/1562 [==============================] - 113s 72ms/step - loss: 0.5856 - acc: 0.8610 - val_loss: 0.6263 - val_acc: 0.8650
Epoch 104/150
1563/1562 [==============================] - 122s 78ms/step - loss: 0.5795 - acc: 0.8637 - val_loss: 0.5892 - val_acc: 0.8736
Epoch 105/150
1563/1562 [==============================] - 116s 74ms/step - loss: 0.5814 - acc: 0.8619 - val_loss: 0.5814 - val_acc: 0.8741
Epoch 106/150
1563/1562 [==============================] - 112s 72ms/step - loss: 0.5723 - acc: 0.8653 - val_loss: 0.6311 - val_acc: 0.8575
Epoch 107/150
1563/1562 [==============================] - 123s 78ms/step - loss: 0.5713 - acc: 0.8634 - val_loss: 0.7330 - val_acc: 0.8270
Epoch 108/150
1563/1562 [==============================] - 120s 77ms/step - loss: 0.5748 - acc: 0.8647 - val_loss: 0.6085 - val_acc: 0.8670
Epoch 109/150
1563/1562 [==============================] - 114s 73ms/step - loss: 0.5678 - acc: 0.8675 - val_loss: 0.5756 - val_acc: 0.8730
Epoch 110/150
1563/1562 [==============================] - 114s 73ms/step - loss: 0.5637 - acc: 0.8666 - val_loss: 0.6420 - val_acc: 0.8569
Epoch 111/150
1563/1562 [==============================] - 116s 74ms/step - loss: 0.6397 - acc: 0.8425 - val_loss: 0.6057 - val_acc: 0.8655
Epoch 112/150
1563/1562 [==============================] - 116s 75ms/step - loss: 0.5961 - acc: 0.8581 - val_loss: 0.6100 - val_acc: 0.8627
```

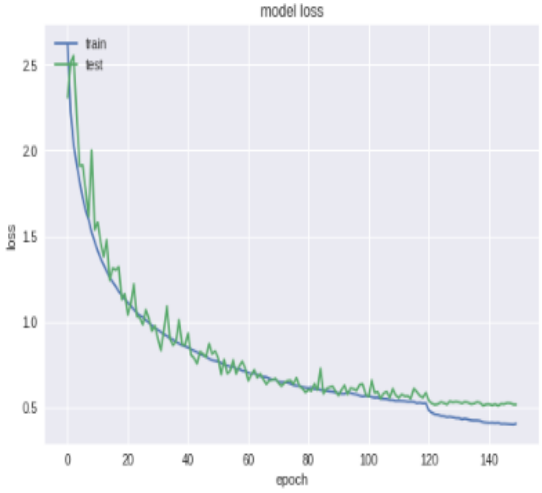
From the above figure, it is predictable that the accuracy might increase above 86%

MODEL 29:

Without Early Stopping :

```
Epoch 130/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4379 - acc: 0.9081 - val_loss: 0.5344 - val_acc: 0.8901
Epoch 131/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4376 - acc: 0.9076 - val_loss: 0.5299 - val_acc: 0.8925
Epoch 132/150
1563/1562 [==============================] - 103s 66ms/step - loss: 0.4311 - acc: 0.9094 - val_loss: 0.5236 - val_acc: 0.8942
Epoch 133/150
1563/1562 [==============================] - 103s 66ms/step - loss: 0.4352 - acc: 0.9075 - val_loss: 0.5328 - val_acc: 0.8912
Epoch 134/150
1563/1562 [==============================] - 103s 66ms/step - loss: 0.4298 - acc: 0.9106 - val_loss: 0.5279 - val_acc: 0.8920
Epoch 135/150
1563/1562 [==============================] - 103s 66ms/step - loss: 0.4250 - acc: 0.9120 - val_loss: 0.5192 - val_acc: 0.8921
Epoch 136/150
1563/1562 [==============================] - 103s 66ms/step - loss: 0.4244 - acc: 0.9116 - val_loss: 0.5223 - val_acc: 0.8920
Epoch 137/150
1563/1562 [==============================] - 103s 66ms/step - loss: 0.4240 - acc: 0.9115 - val_loss: 0.5317 - val_acc: 0.8895
Epoch 138/150
1563/1562 [==============================] - 103s 66ms/step - loss: 0.4221 - acc: 0.9119 - val_loss: 0.5265 - val_acc: 0.8928
Epoch 139/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4144 - acc: 0.9146 - val_loss: 0.5105 - val_acc: 0.8964
Epoch 140/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4115 - acc: 0.9154 - val_loss: 0.5176 - val_acc: 0.8957
Epoch 141/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4103 - acc: 0.9158 - val_loss: 0.5187 - val_acc: 0.8958
Epoch 142/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4109 - acc: 0.9158 - val_loss: 0.5128 - val_acc: 0.8967
Epoch 143/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4079 - acc: 0.9162 - val_loss: 0.5188 - val_acc: 0.8955
Epoch 144/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4110 - acc: 0.9149 - val_loss: 0.5102 - val_acc: 0.8964
Epoch 145/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4038 - acc: 0.9175 - val_loss: 0.5206 - val_acc: 0.8946
Epoch 146/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4045 - acc: 0.9161 - val_loss: 0.5190 - val_acc: 0.8949
Epoch 147/150
1563/1562 [==============================] - 102s 66ms/step - loss: 0.4029 - acc: 0.9169 - val_loss: 0.5256 - val_acc: 0.8938
Epoch 148/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.4012 - acc: 0.9172 - val_loss: 0.5249 - val_acc: 0.8932
Epoch 149/150
1563/1562 [==============================] - 102s 65ms/step - loss: 0.3998 - acc: 0.9185 - val_loss: 0.5174 - val_acc: 0.8945
Epoch 150/150
1563/1562 [==============================] - 101s 65ms/step - loss: 0.4045 - acc: 0.9164 - val_loss: 0.5166 - val_acc: 0.8942
```



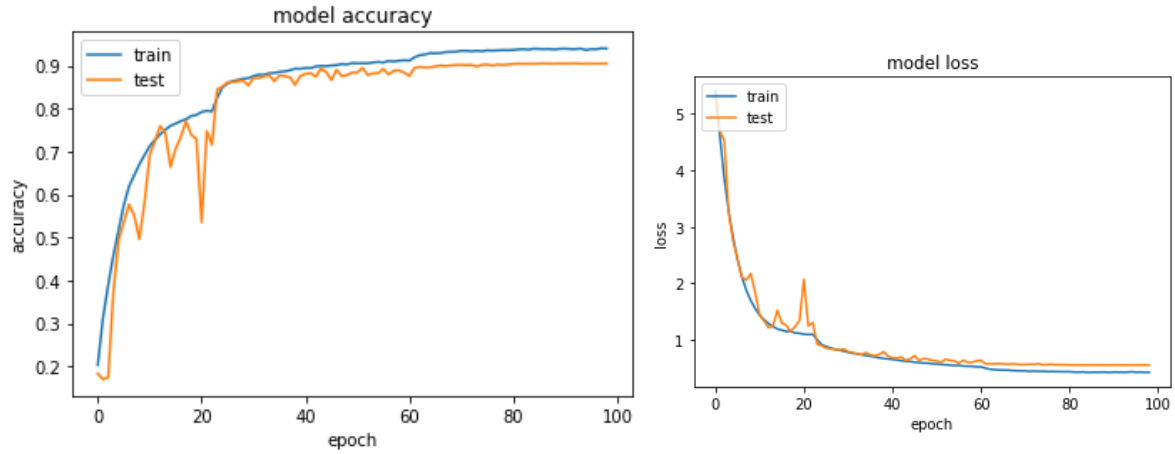


This model has given as a very good accuracy of Training – 91% and test of 89%.This model though has learned slowly, performed in a better way.

This model will be used as final model (1) in this project.

MODEL 30 :
Combining all the results together,
With weight decay = 0.01 ,
Dropout = 0.35
Learning rate = 0.015
Momentum = 0.9
With ReduceLROnPlateau as callback
This model gave us a training accuracy of 93% and test accuracy of 90%.

```
Epoch 82/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4369 - acc: 0.9378 - val_loss: 0.5592 - val_acc: 0.9044
Epoch 83/200
196/196 [==============================] - 18s 90ms/step - loss: 0.4353 - acc: 0.9381 - val_loss: 0.5607 - val_acc: 0.9041
Epoch 84/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4391 - acc: 0.9371 - val_loss: 0.5591 - val_acc: 0.9042
Epoch 85/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4308 - acc: 0.9394 - val_loss: 0.5607 - val_acc: 0.9040
Epoch 86/200
196/196 [==============================] - 15s 79ms/step - loss: 0.4330 - acc: 0.9386 - val_loss: 0.5598 - val_acc: 0.9044
Epoch 87/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4338 - acc: 0.9383 - val_loss: 0.5604 - val_acc: 0.9047
Epoch 88/200
196/196 [==============================] - 15s 79ms/step - loss: 0.4344 - acc: 0.9385 - val_loss: 0.5599 - val_acc: 0.9045
Epoch 89/200
196/196 [==============================] - 16s 82ms/step - loss: 0.4356 - acc: 0.9376 - val_loss: 0.5596 - val_acc: 0.9044
Epoch 90/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4313 - acc: 0.9390 - val_loss: 0.5592 - val_acc: 0.9046
Epoch 91/200
196/196 [==============================] - 15s 79ms/step - loss: 0.4364 - acc: 0.9394 - val_loss: 0.5604 - val_acc: 0.9045
Epoch 92/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4349 - acc: 0.9384 - val_loss: 0.5605 - val_acc: 0.9046
Epoch 93/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4342 - acc: 0.9381 - val_loss: 0.5594 - val_acc: 0.9049
Epoch 94/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4350 - acc: 0.9395 - val_loss: 0.5604 - val_acc: 0.9044
Epoch 95/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4422 - acc: 0.9360 - val_loss: 0.5606 - val_acc: 0.9042
Epoch 96/200
196/196 [==============================] - 15s 79ms/step - loss: 0.4323 - acc: 0.9382 - val_loss: 0.5594 - val_acc: 0.9045
Epoch 97/200
196/196 [==============================] - 15s 77ms/step - loss: 0.4375 - acc: 0.9382 - val_loss: 0.5603 - val_acc: 0.9045
Epoch 98/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4312 - acc: 0.9400 - val_loss: 0.5598 - val_acc: 0.9044
Epoch 99/200
196/196 [==============================] - 15s 78ms/step - loss: 0.4322 - acc: 0.9397 - val_loss: 0.5600 - val_acc: 0.9046
```



This is our final model (2) which will also be used for this project.