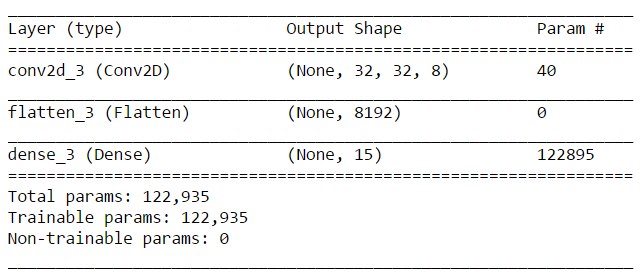
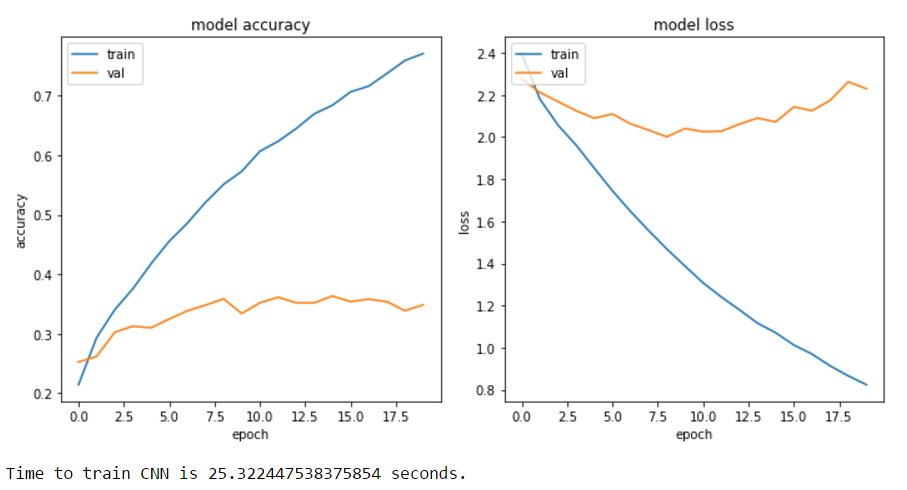


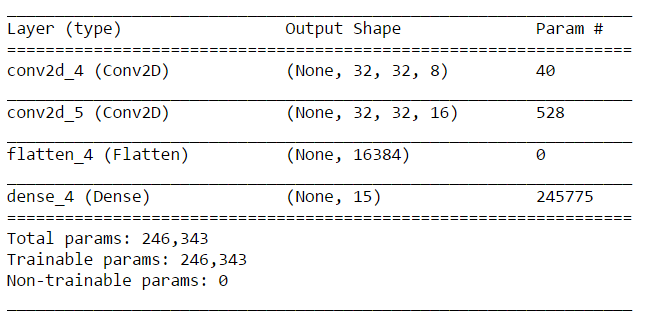
1. Start with simple model of one CNN and one FC layer

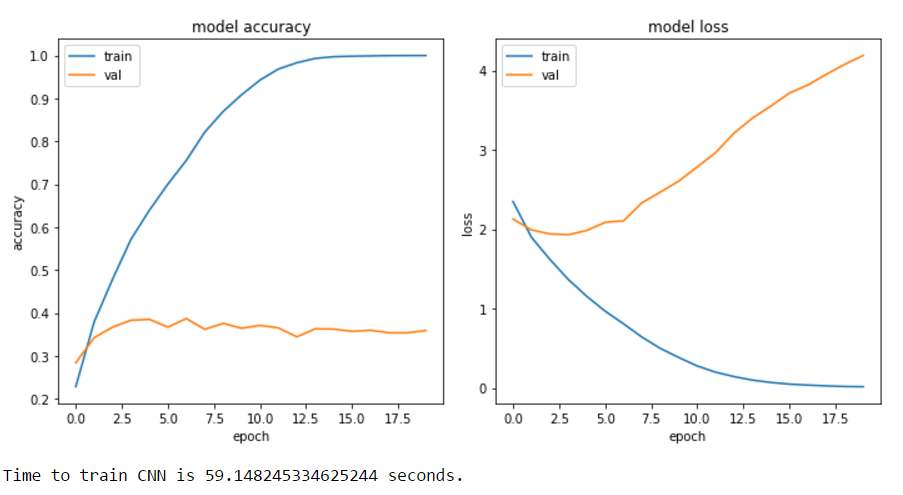




Test accuracy of the model is around 35%. This already better than the test accuracy of my benchmark model, SVM. The model does great on training set but performs very poorly on validation set.

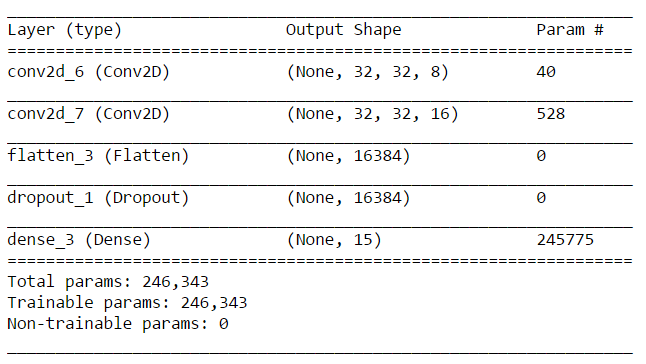
1. Adding one more convolution layer so that the model learns more features. Two CNN and one FC layer

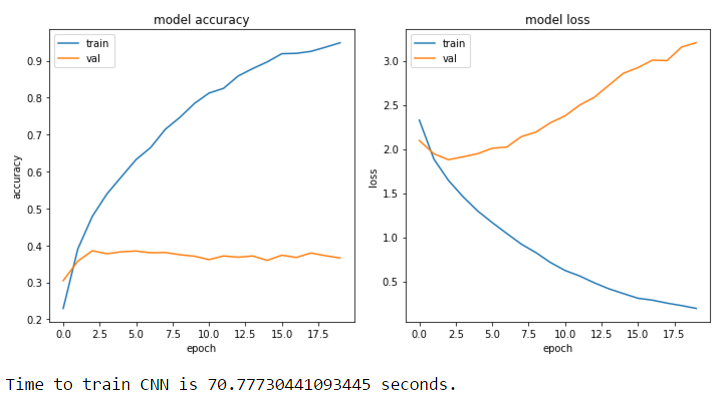




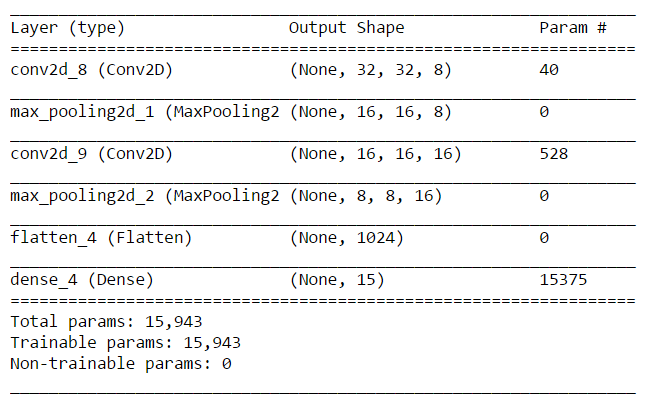
Looking at the training curve, the model now seems to learn but the model is overfitting as shown by validation curve.The overfitting can be reduced by adding non-linear layers like droupout or max pooling

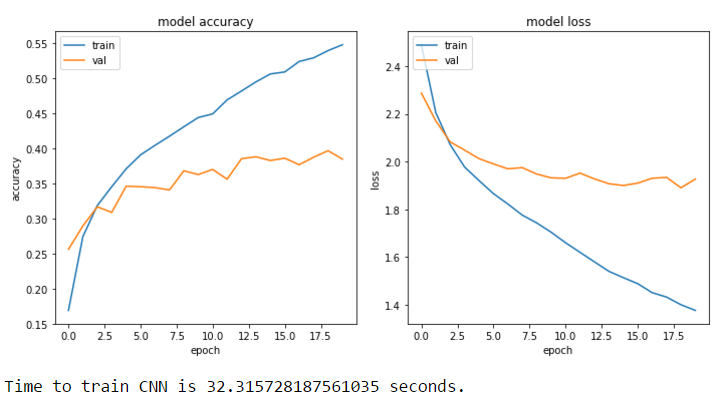
1. Adding dropout layer



  
There is no improvement. Now let’s try with maxpooling layer instead of dropout.

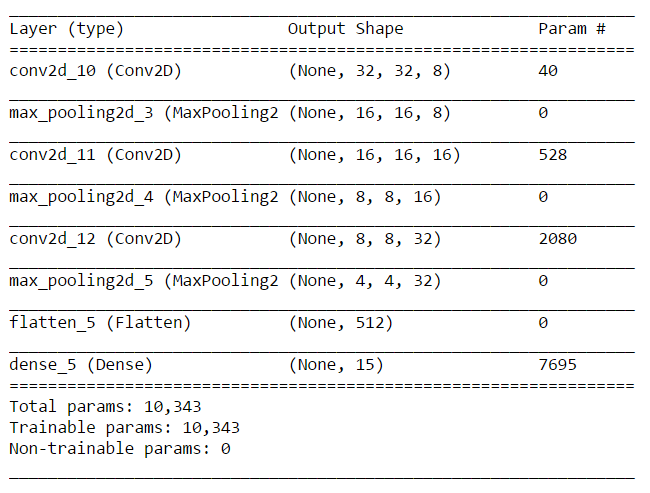
1. Adding maxpooling layer

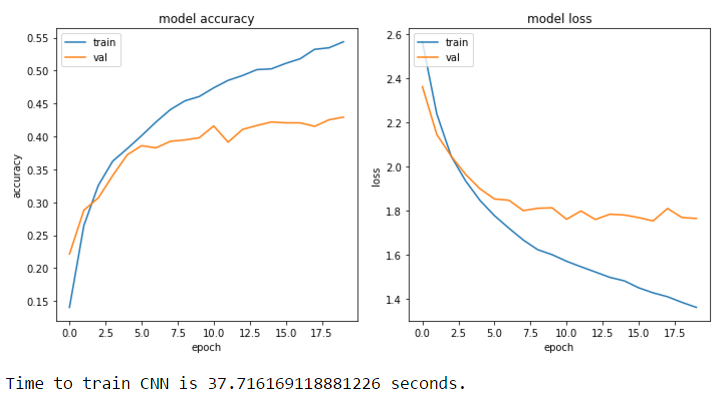




The maxpooling layer has removed the overfitting but the training curve is performing poorly. Let’s check if the model does any good by adding another convolution layer so that it can learn more features

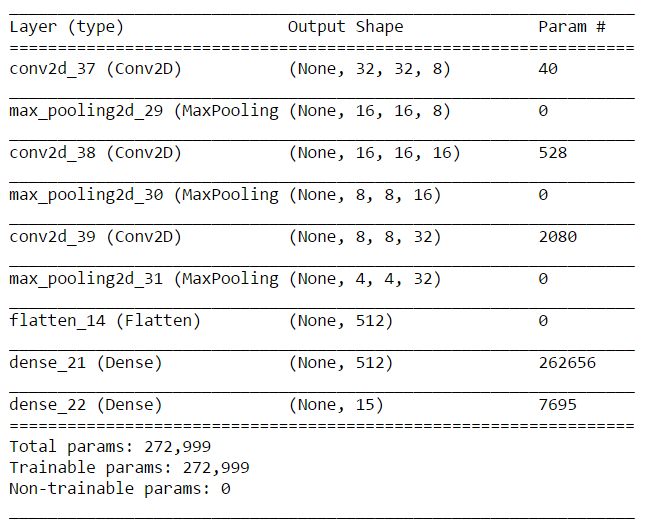
1. Adding another convolution layer.

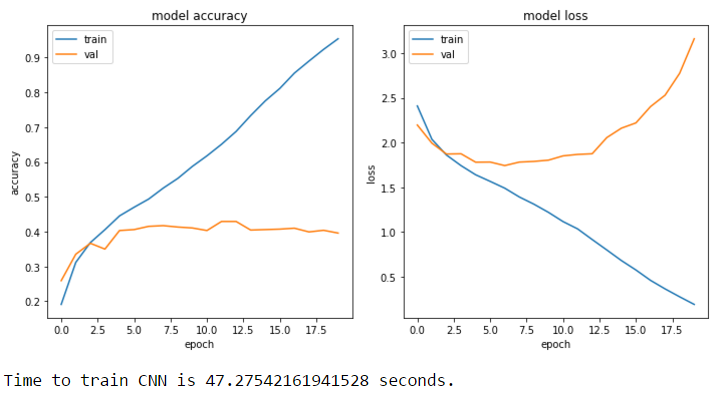




Adding another convolution layer seems to have improved the model performance on validation set but there is not much improvement on training set. Testing accuracy of the model is 42.7333%. Let’s now add one more dense layer to check if training improves.

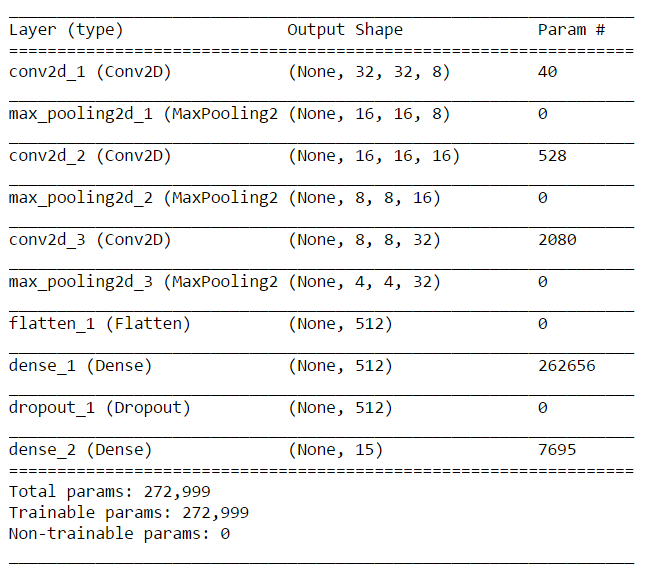
1. Adding a dense layer

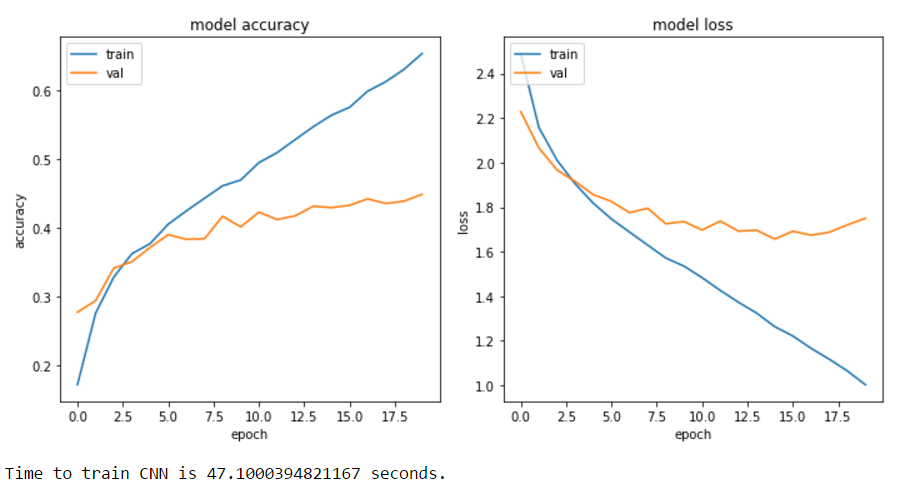




As seen, adding dense layer has increased the accuracy of training set and slightly reduced accuracy on validation set. Also the model is overfitting on validation set. Check now by adding dropout on dense layers

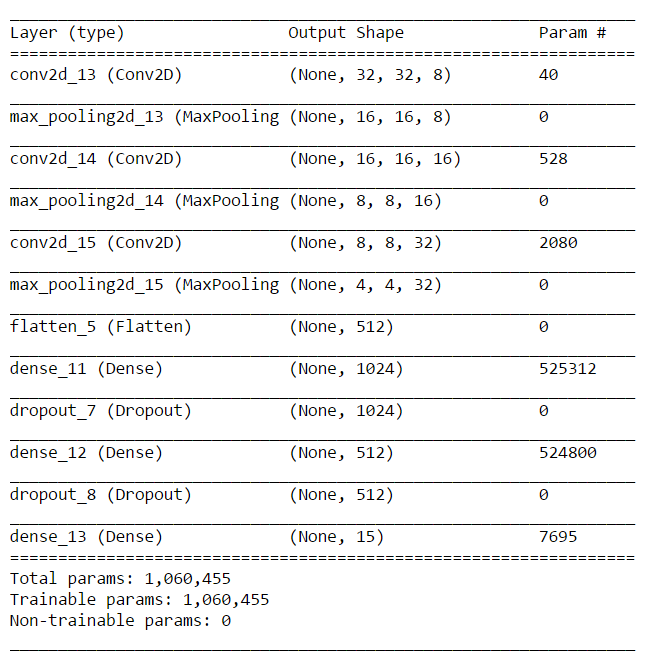
1. Adding dropout with 0.5

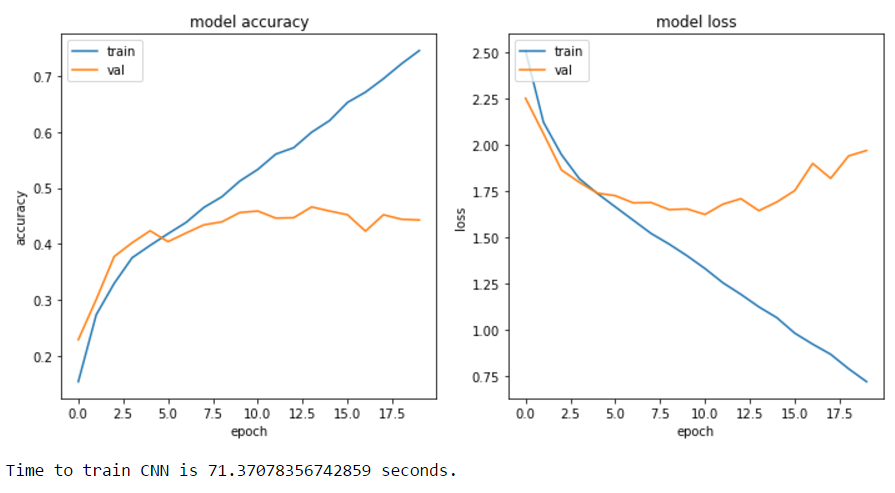




Accuracy is 45.13%. The overfitting on validation set is reduced but at the cost of training accuracy. Add more dense layers to improve training set

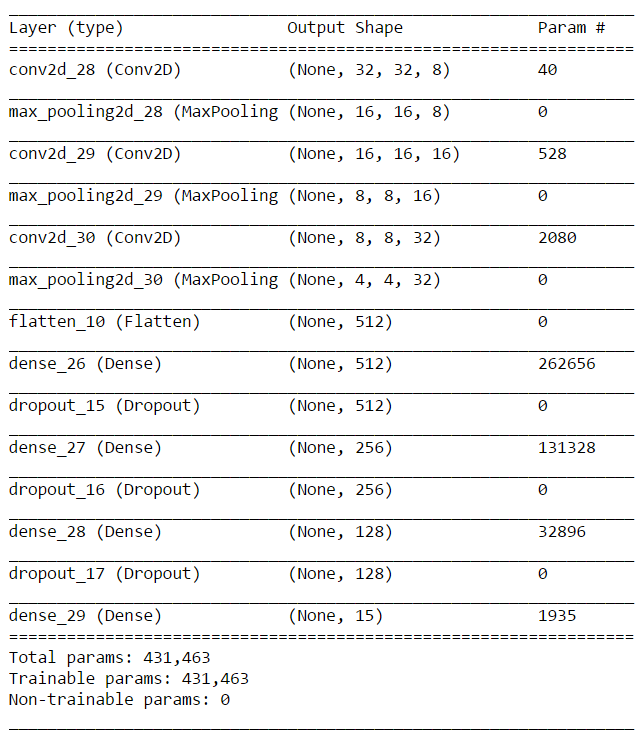
1. Add one more dense layer with dropout 0.5

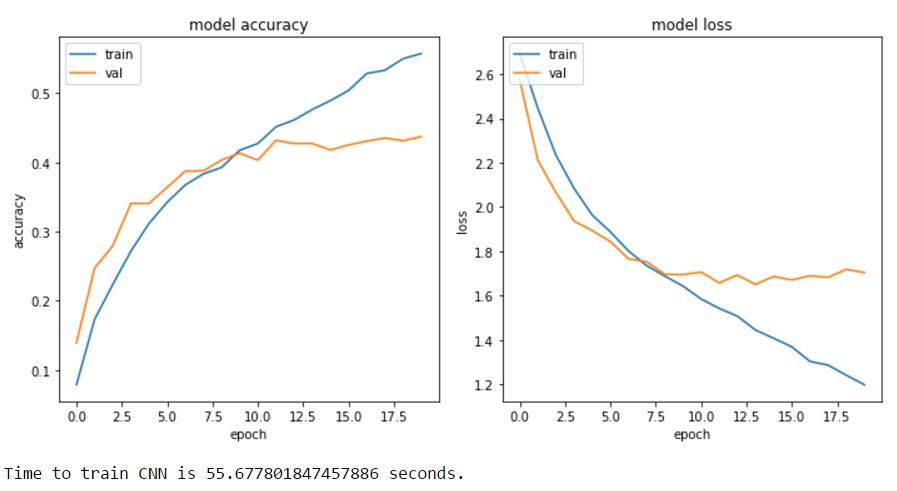




Accuracy is 44.27%. The test accuracy reduced with no significant improvement in training curve.

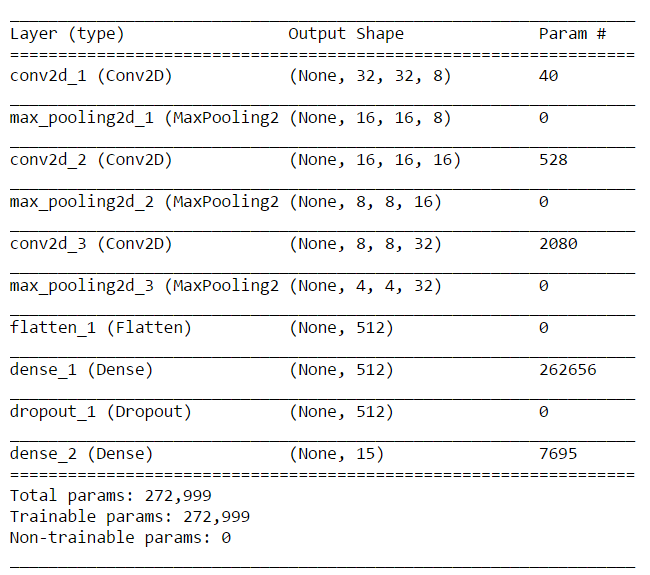
1. Add one more dense with dropout 0.5





Accuracy is 45.13%. Adding more dense layer doesn’t make a significant difference. So going back to model with 2 dense layers

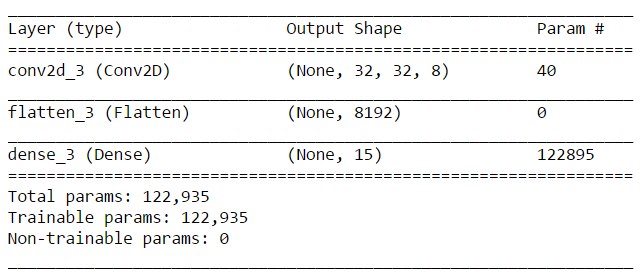
1. Final model



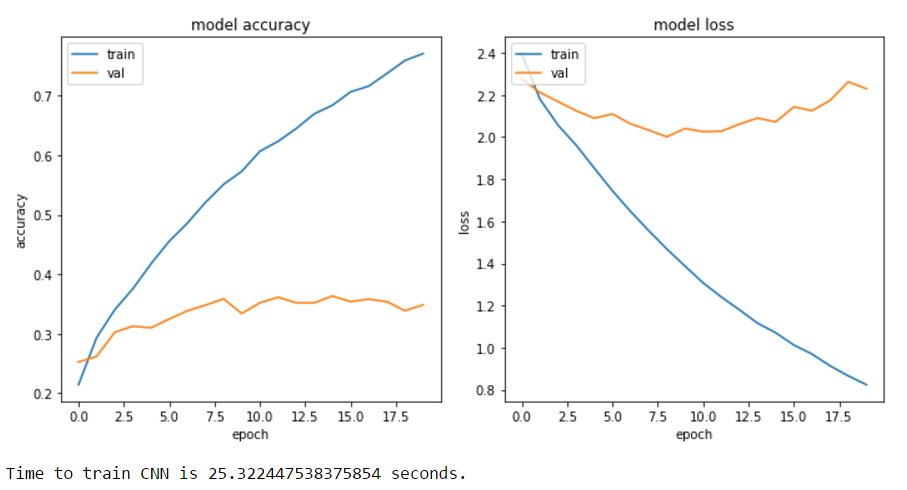


Test accuracy is 45.27%

1. Model 1: My initial model had one convolution layer and one fully connected layer:

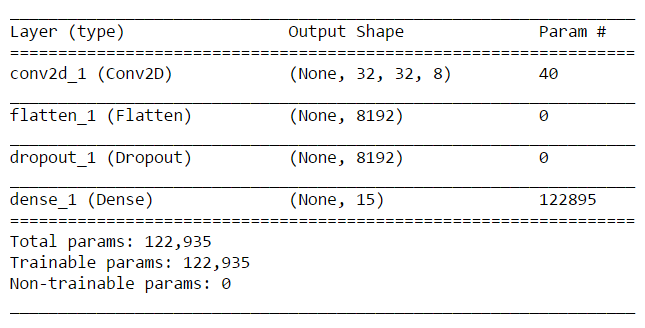


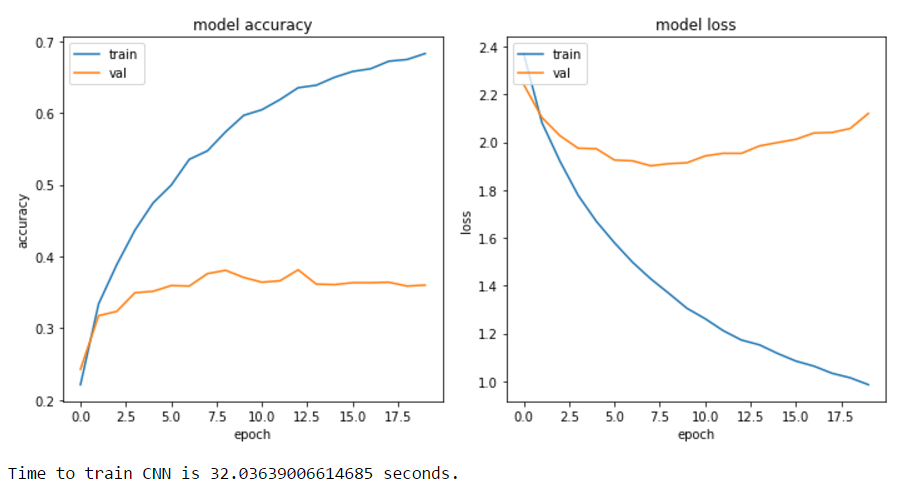
The test accuracy of this model is around 35% which is already better than the test accuracy of the benchmark model. Below is the performance of the model across 20 iterations:



Though the accuracy of the model is better than that of the benchmark model, the graphs above tell a different story. The loss on validation set seems to decrease for the first few iterations and then gradually increase. This is a typical case of overfitting. Overfitting can be reduced by introducing non-linearity in the model. Keras provides non-linear layers such as dropout and max-pooling.

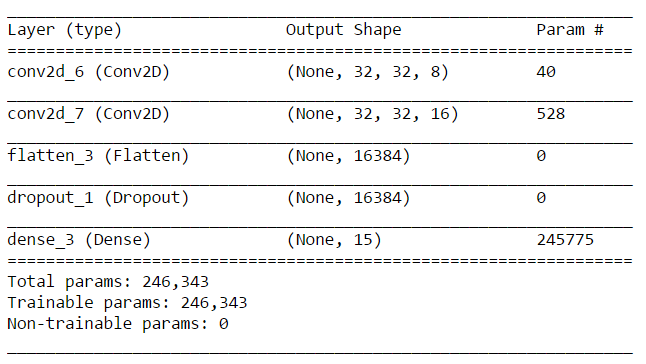
1. Model 2: Added dropout layer with rate 0.5 to reduce overfitting:

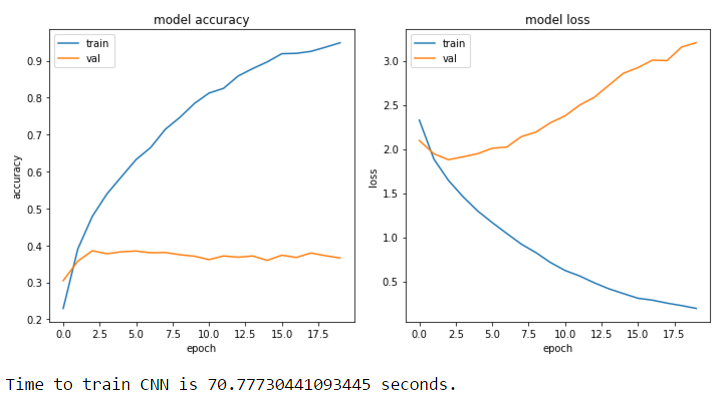




Adding dropout layer reduced the overfitting only slightly. The model needs more features to learn better.

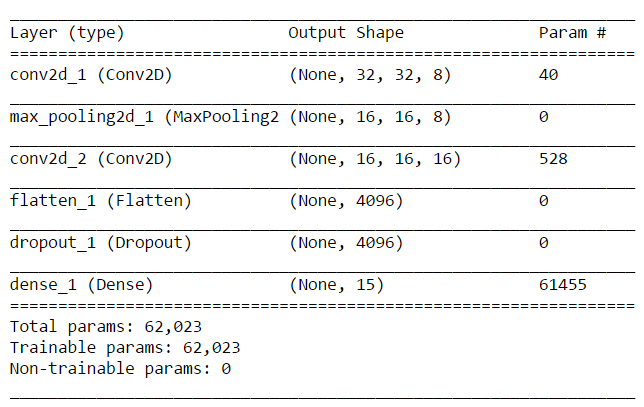
1. Model 3: To help the model learn better, I added one more convolution layer through which the model can extract more features from the image:

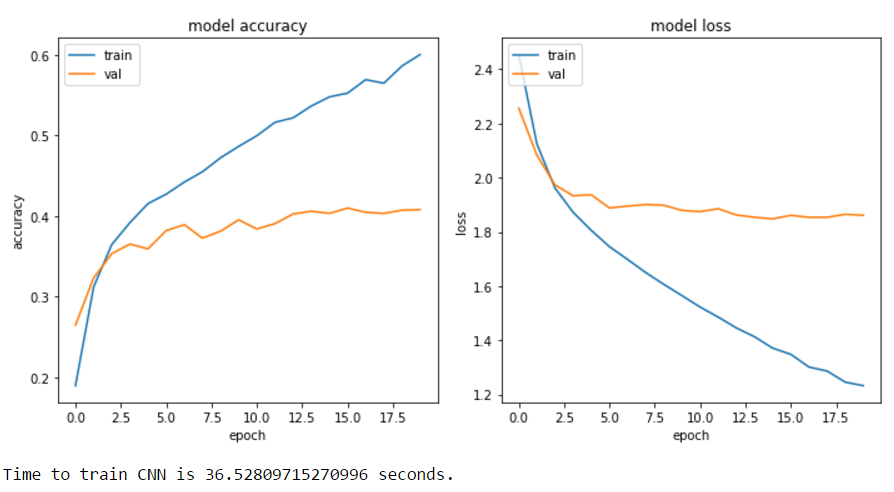




Adding one more convolution layer has increased the number of model parameters leading to increase in training time. The model is doing good on the training dataset but the it is overfitting as shown by validation curve.

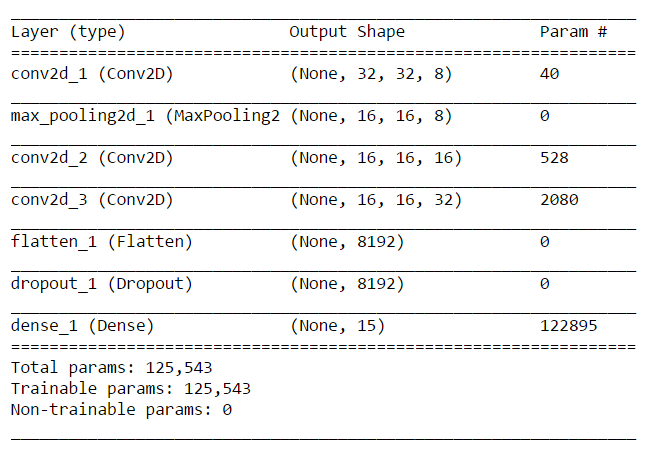
1. Model 4: Now added max-pooling layer to reduce overfitting:

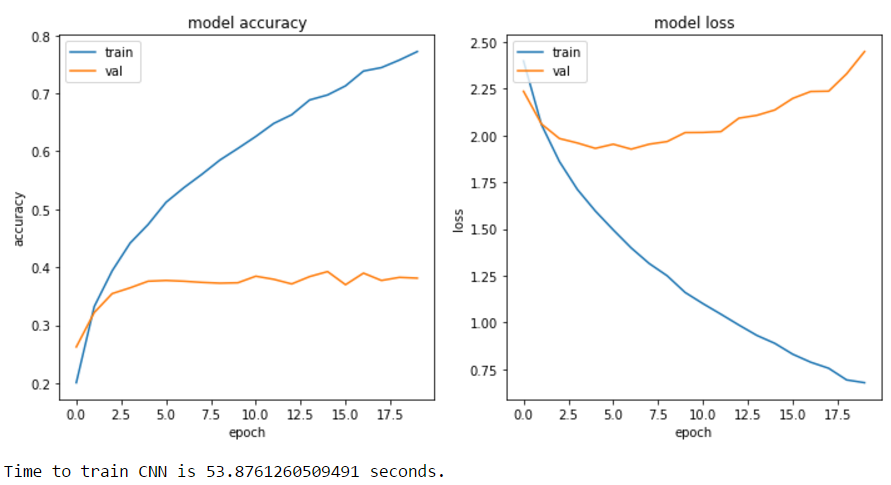




The max-pooling layer was indeed successful in reducing overfitting but at the cost of training accuracy.

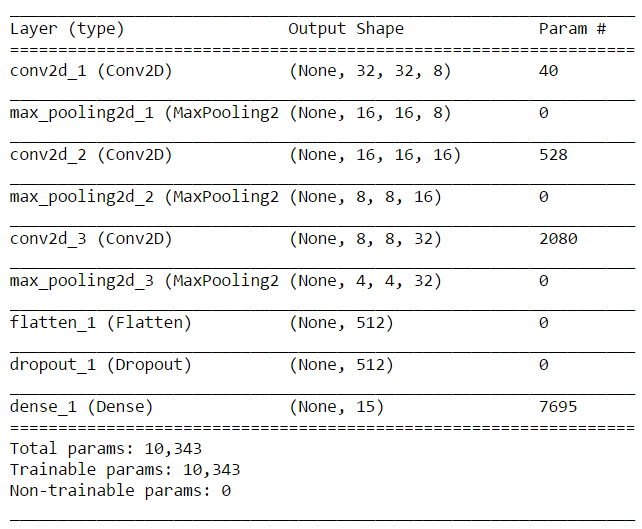
1. Model 5: Added another convolution layer in hope that the model learns more feature:

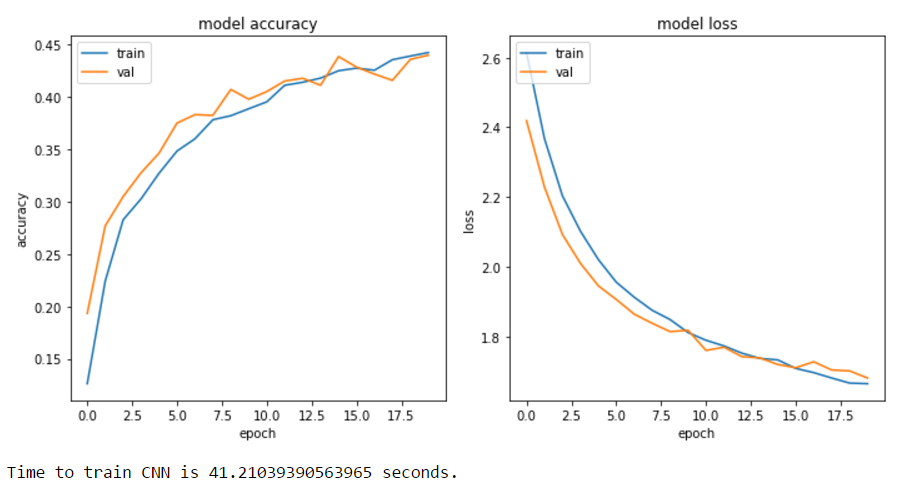




Adding another convolution layer has improved the training curve but the model is overfitting again. Test accuracy of this model is 38.26%.

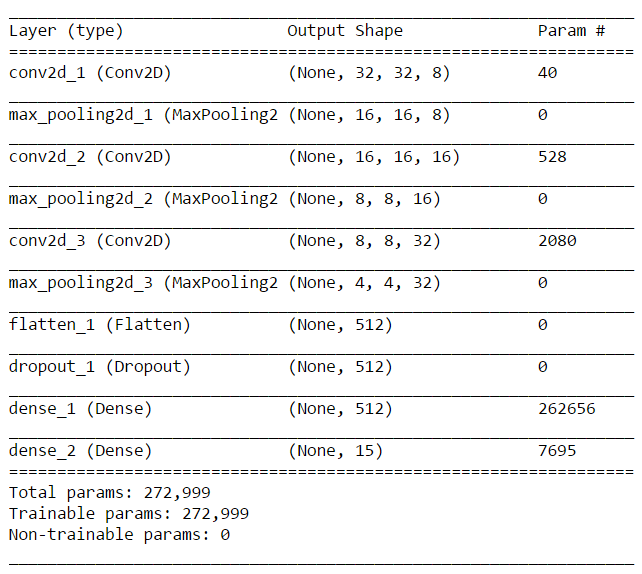
1. Model 6: Added max-pooling layers after convolution layers to reduce overfitting:

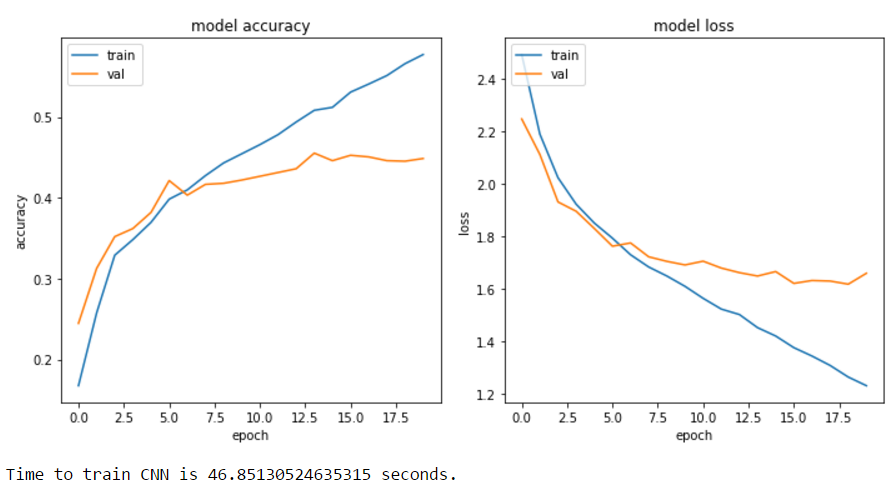




The max-pooling layers did a good job of eliminating overfitting but it has drastically reduced the training accuracy. Test accuracy of this model is 43.93%

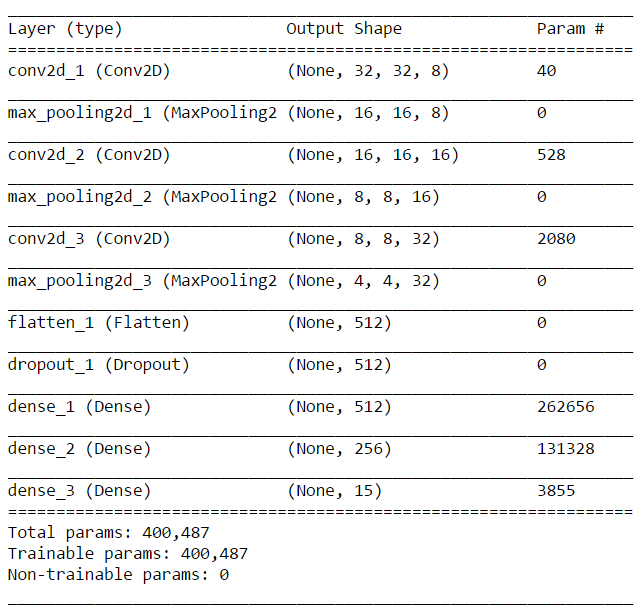
1. Model 7: Added a dense layer to see if it helps:

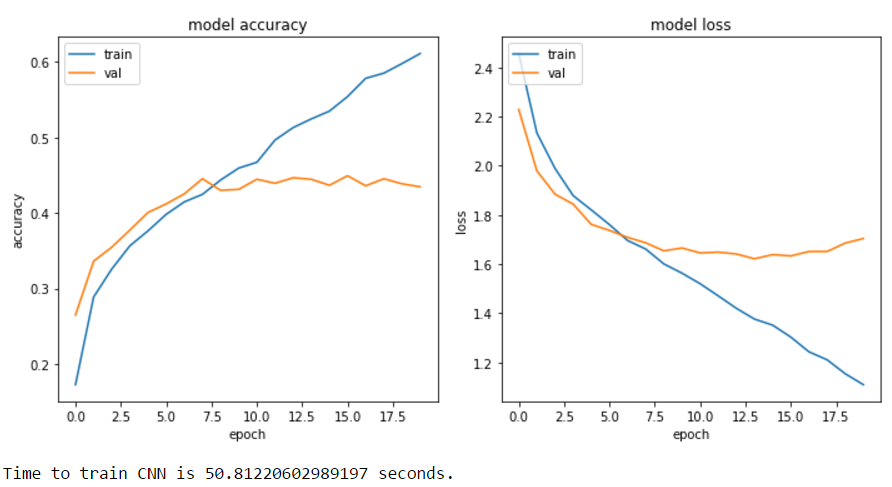




Adding dense layer has a slight improvement in training curve. Test accuracy is 46%.

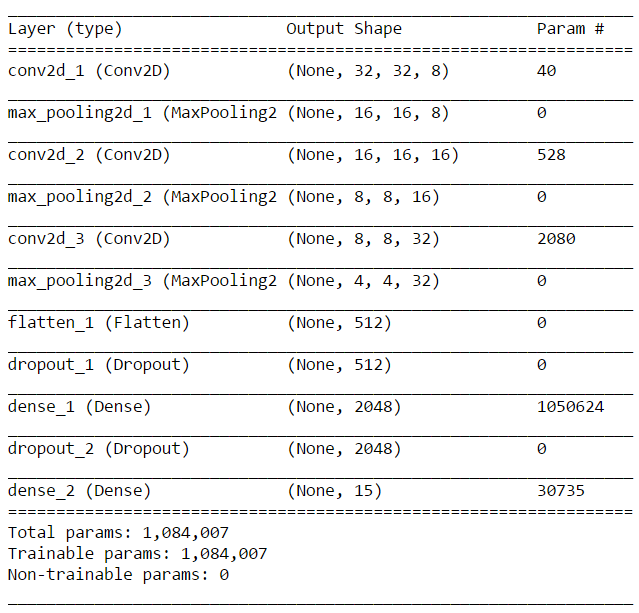
1. Model 8: Added one more dense layer in hope of further improving the training curve:





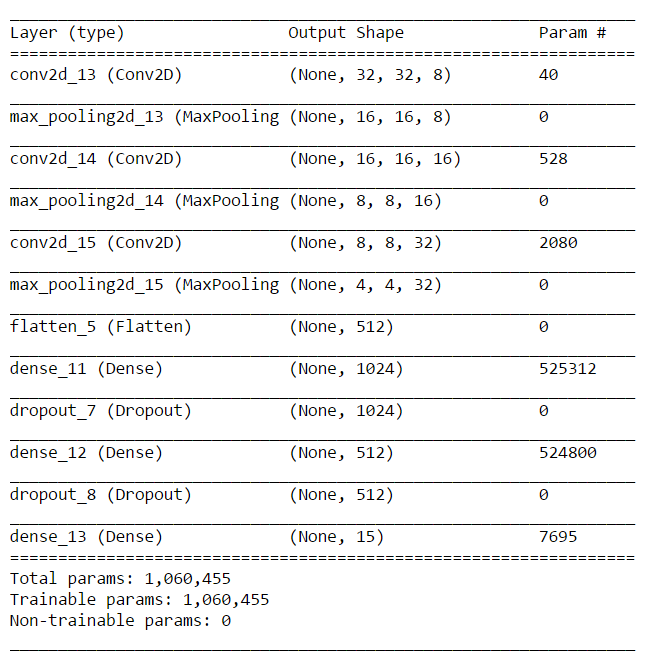
Adding one more dense layer has only a very small improvement on the training curve and also the model has started to over fit. So going back to Model 7.

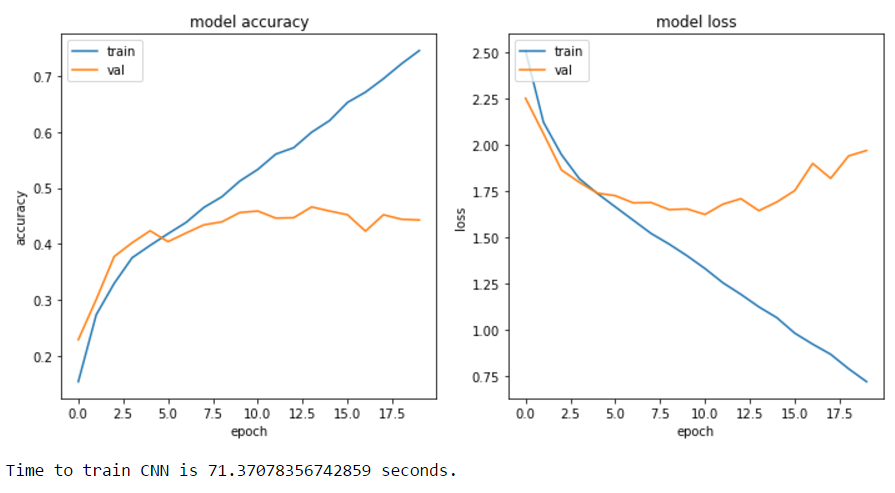
1. Model 9: On Model 7, increased the number of neurons from 512 to 2048 on the dense layer:



Addition of dropout layer has reduced the overfitting but at the cost of training accuracy. Test accuracy of this model is 45.13%.

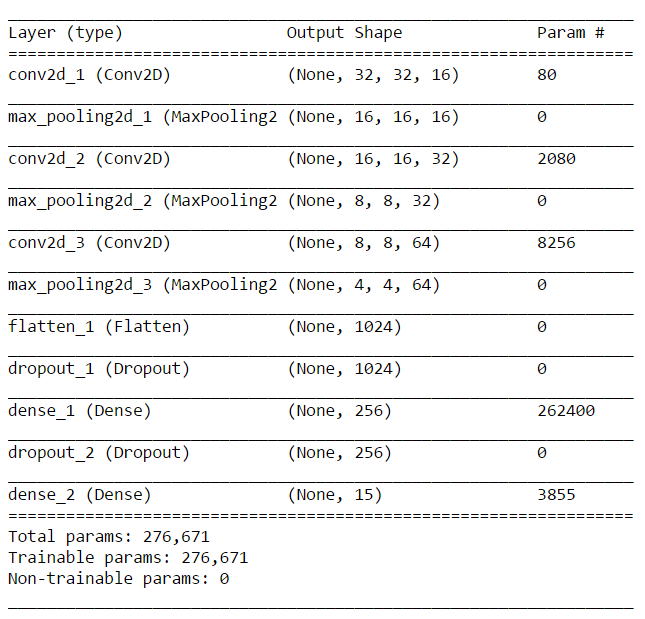
1. Model 8: Added one more dense layer to improve training curve along with dropout layer to reduce overfitting:

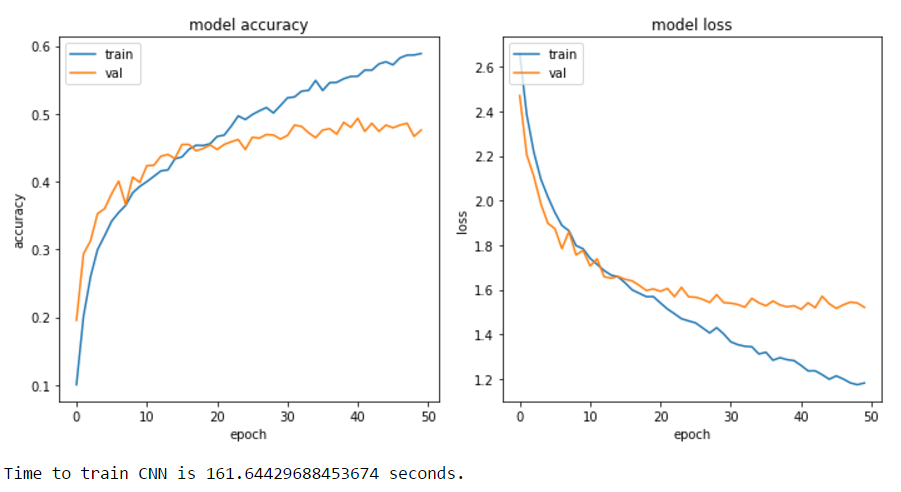




There is slight improvement in training curve but the model is overfitting. Also with test accuracy reduced to 44.27%

1. Model 9:





First dropout rate 0.5 and then 0.5

EPOCHS 50: 49.73%

1. Model 10:

50.2 sec and 43.8% with -0.5 to 0.5

50.3 sec and 43.9%

50.3 and 44.7%

50.3 and 44.5%

50.5 and 45.3% with 0 to 1

50.04 and 43.26%

50.4 and 44.3%

50.6 and 43.3