Name: Hithesh Shanmugam

Assignment 7

CSC 578

Kaggle user name: Hithesh Shanmugam

Best Model: (Kaggle Score: 0.67755 (public score), 0.75417 (private score)

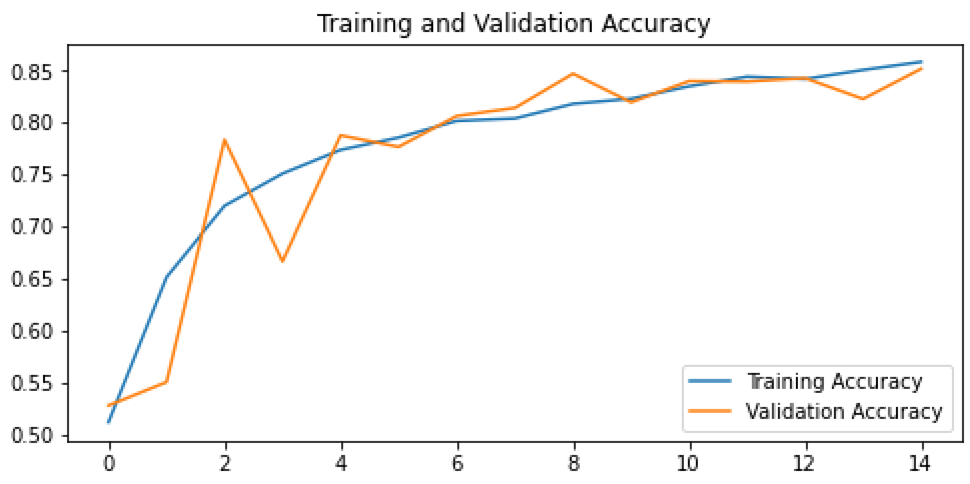
In this assignment I tried to construct different convolution neural network models by adjusting the hyperparameters of the CNNs and every model was trained by 15 epochs in that my best model turned out to be the one where I used the Batch Normalization and spatial dropout and also more dense layers and the total number of parameters can be seen as shown in below which was taken by summarising the model

```
Total params: 2,400,230
Trainable params: 2,397,798
Non-trainable params: 2,432
```

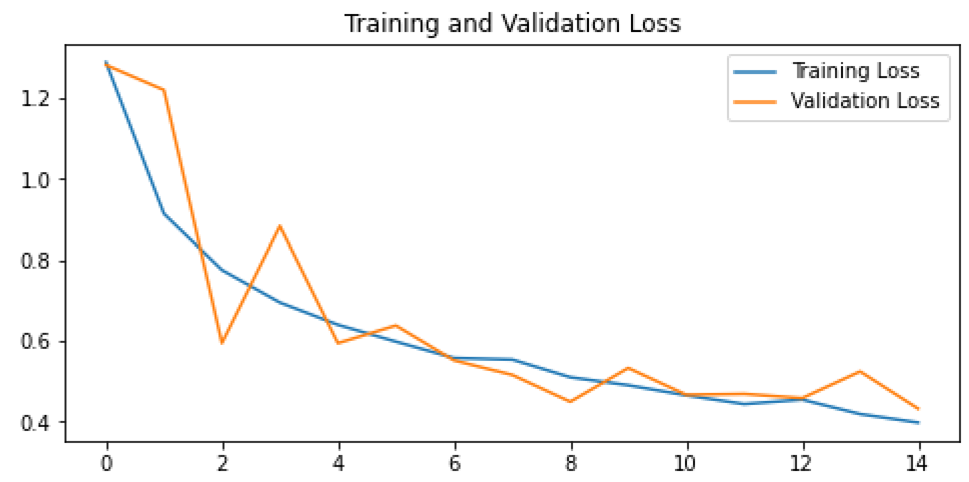
The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 22s 63ms/step - loss: 0.3973 - accuracy: 0.8583 - val_loss: 0.4317 - val_accuracy: 0.8516
```

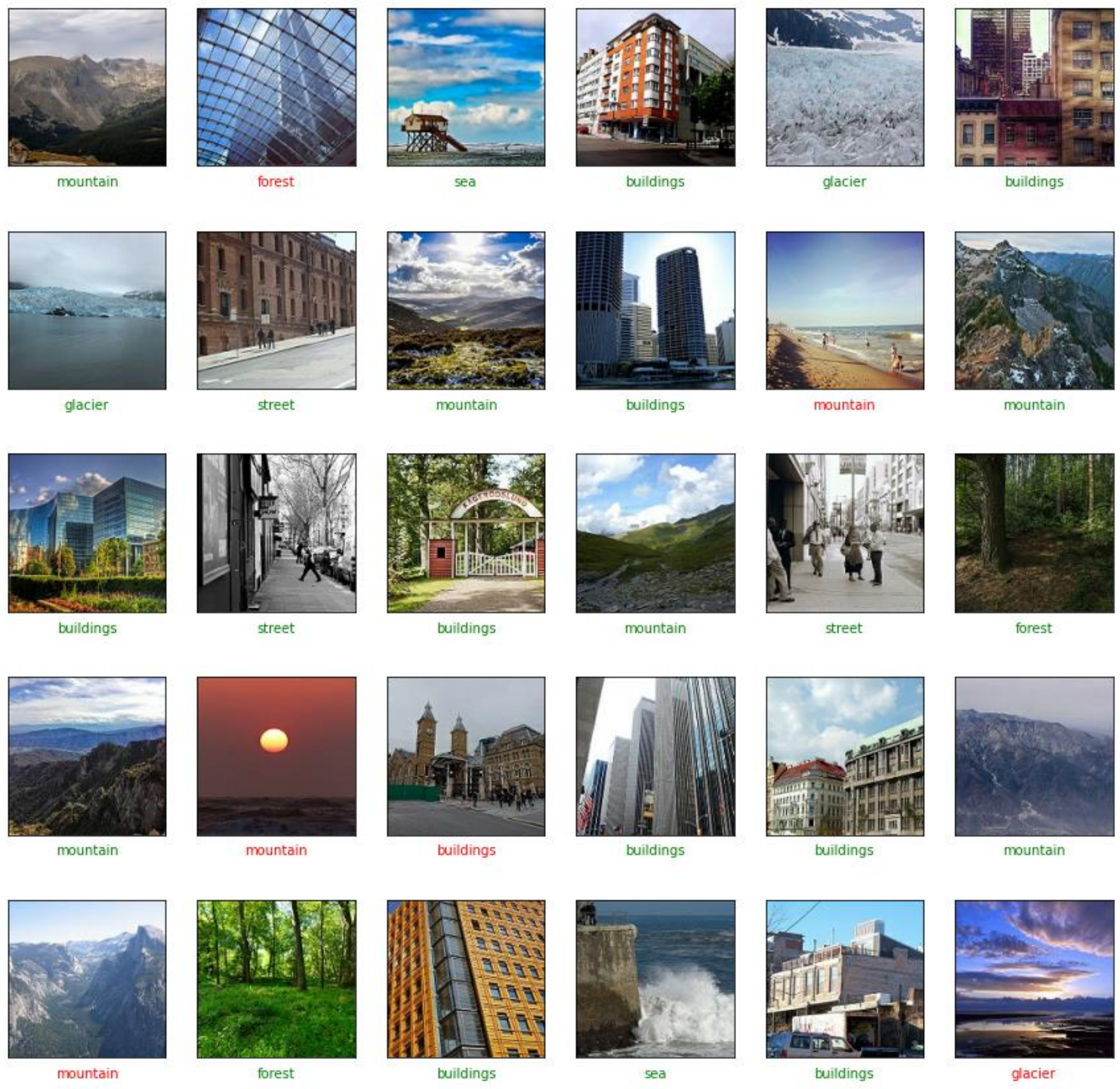
The visual representation of the training and validation accuracy can be seen in the picture



The visual representation of the training and validation loss can be seen in the picture



The visual representation of how the model predicts the for 30 images can be seen below



As we can see in the picture there were six wrong predictions here but there were models which I constructed had fewer wrong images than this model but it didn't have the better result than this model as we know there are total of 2993 images in the data. And how I got this model will be explained at the model number 11.

I constructed a total of 11 models and I tuned the hyper parameters with respect to additional layers, padding, batch normalization, spatial dropout.

After explaining all the model, I have shown the comparison of every model and let's call the given model as original model.

Model 1: Adding additional layer

Expectation- As the number of layer increases in the network the prediction would be better so by adding a single additional layer, I thought that the model would perform better than the fewer layers model.

Result- As I expected the model predicted better than the fewer layer model but note that it didn't give me a big difference. For 30 images the original model predicted six wrong images and this model predicted five wrong images.

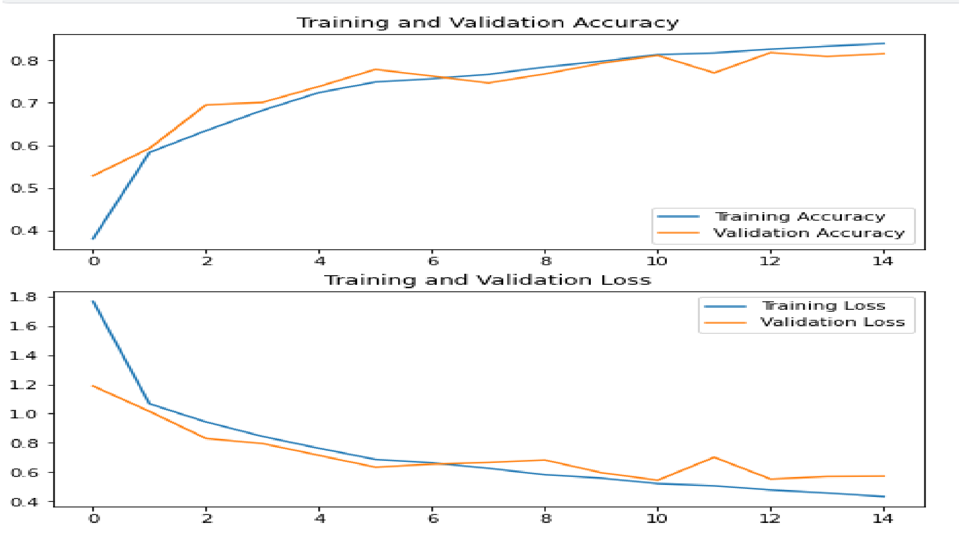
The total number of parameters for this model is

```
=================================================================
Total params: 112,070
Trainable params: 112,070
Non-trainable params: 0
_________________________________________________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 12s 33ms/step - loss: 0.4339 - accuracy: 0.8392 - val_loss: 0.5724 - val_accuracy: 0.8155
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



Model 2: Removing the layers

Expectation- We know that lesser the layer in the CNN prediction is not as good as more layers so by removing the layers I can tell that it is not going to be better so I added padding to the layers so I expected adding padding to the image processed by a CNN allows for more accurate analysis of images.

Result- By adding padding to the fewer layers in the model it was not the best it predicted in the end it turned out to be my one of the worst models I have created it. The reason I have gone for lesser layers is to work faster but the result was not I expected.

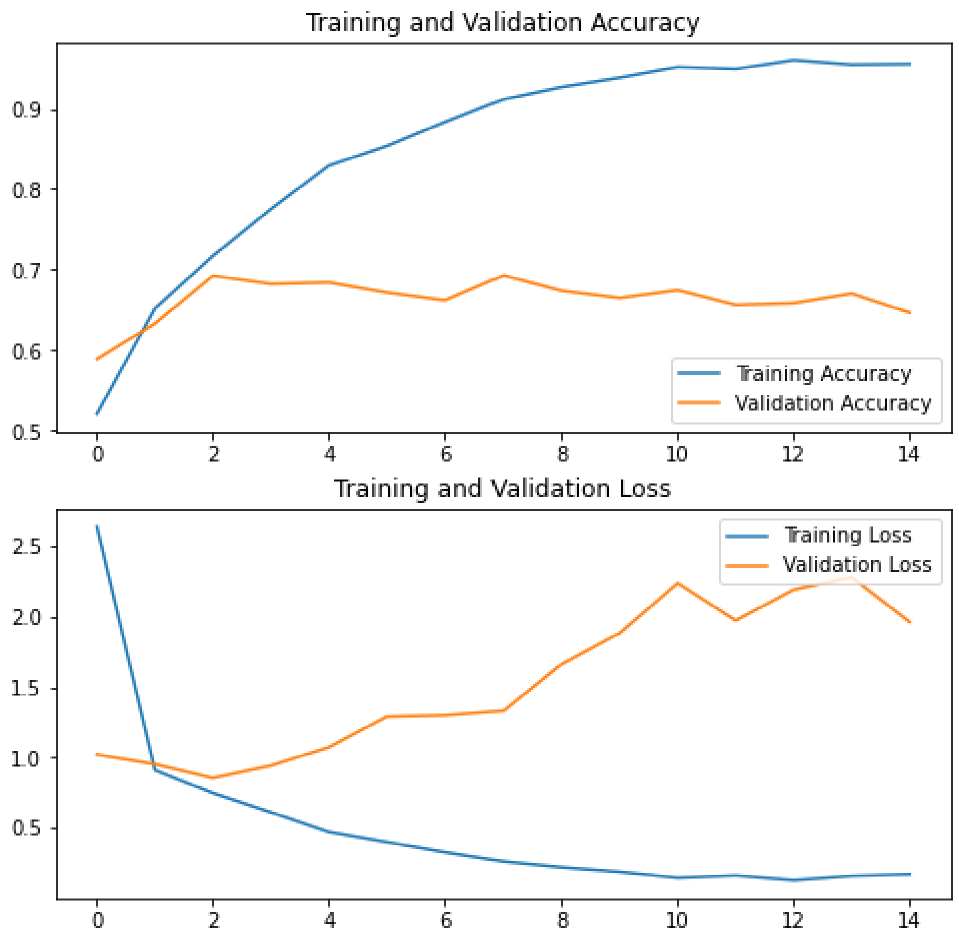
The total number of parameters for this model is

```
==============================
Total params: 2,590,406
Trainable params: 2,590,406
Non-trainable params: 0
______________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 11s 31ms/step - loss: 0.1701 - accuracy: 0.9552 - val_loss: 1.9646 - val_accuracy: 0.6468
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



Model 3: Random Selection

Expectation- Since my previous model didn't perform well this time, I selected the layers randomly and also, I used striding in the pooling layers sine it can reduce spatial resolution leading to computational benefits so I expected my model to perform well than the previous model with just padding.

Result- The results came out to be as much as I predicted it performed better than the previous model such that for 30 images this model predicted seven wrong images where as the previous model predicted ten wrong images.

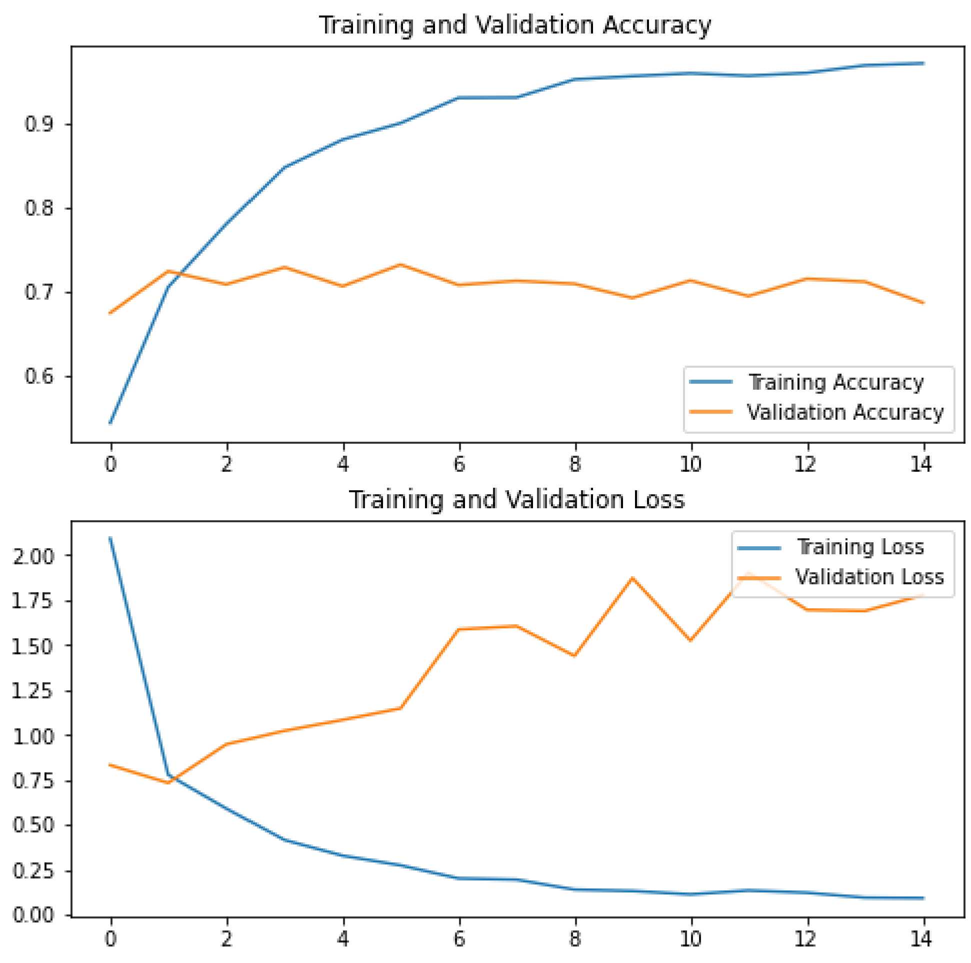
The total number of parameters for this model is

```
Total params: 2,618,150
Trainable params: 2,618,150
Non-trainable params: 0
_________________________________________________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 12s 35ms/step - loss: 0.0917 - accuracy: 0.9723 - val_loss: 1.7786 - val_accuracy: 0.6868
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



Model 4: Adjusting the layers

Expectation- This time instead of choosing random layers I adjusted the layers in a orderly manner and also introduced spatial dropout in the model which is same as regular dropout but here it drops entire 2D feature maps instead of individual elements which helps us in in the problem of overfitting so I expected it to perform better than other previous models.

Result-The result was just as I expected the problem of overfitting was reduced in this model and careful selection of the layers gave me this model to be my second-best model where it got only four wrong predictions in the 30 images.

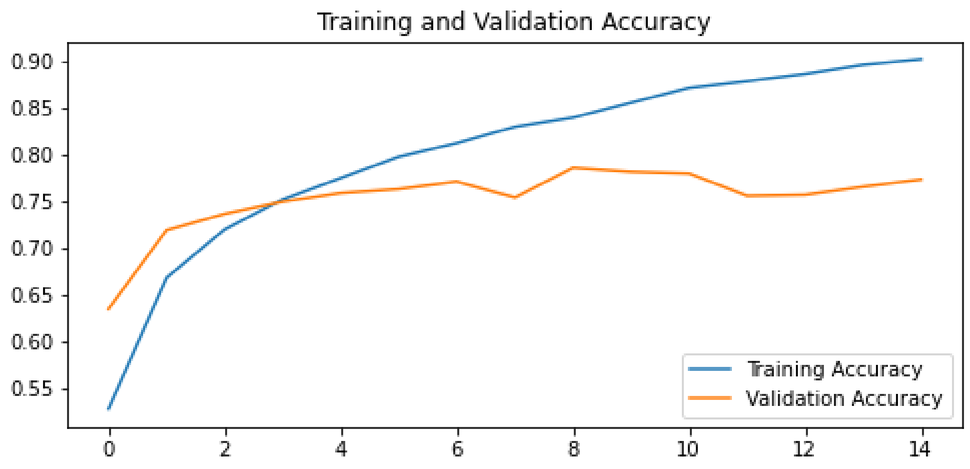
The total number of parameters for this model is

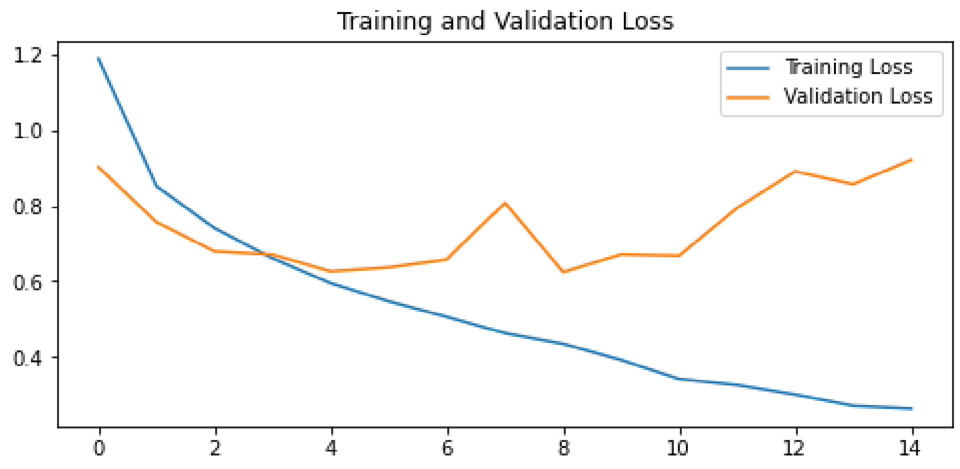
```
========================================
Total params: 338,790
Trainable params: 338,790
Non-trainable params: 0
________________________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 16s 45ms/step - loss: 0.2640 - accuracy: 0.9021 - val_loss: 0.9201 - val_accuracy: 0.7733
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture





Model 5: Additional layers to the adjusted layer model

Expectation- Previously adding additional layer to the original model gave me better result than the original model so I added some extra layers to the best predicted model so far between the models I created so I expect it to perform better than the previous model.

Result- The result surprised me this time because the model didn't perform better than the previous model but it wasn't performing better. Though it predicted the same number of wrong images in the 30 images i.e., 4 wrong images but in Kaggle it wasn't better.

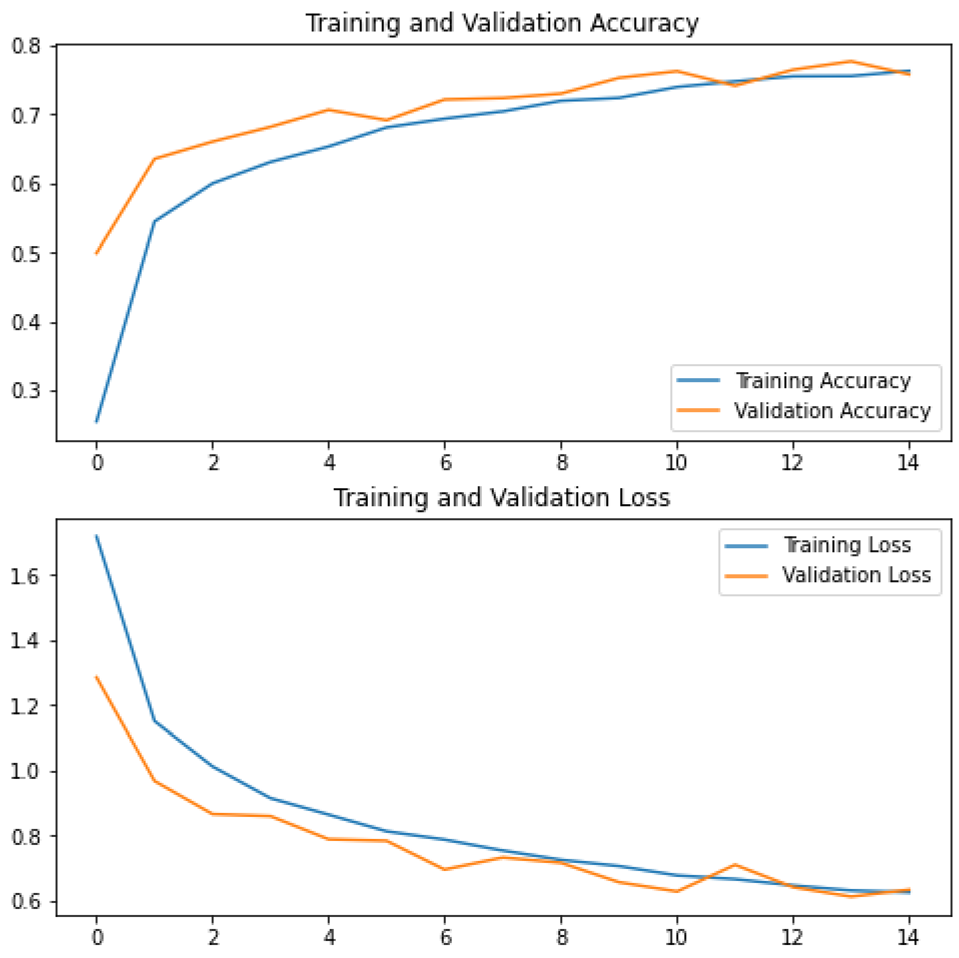
The total number of parameters for this model is

```
=================================================================
Total params: 120,774
Trainable params: 120,774
Non-trainable params: 0
_________________________________________________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 15s 43ms/step - loss: 0.6259 - accuracy: 0.7630 - val_loss: 0.6340 - val_accuracy: 0.7580
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



Model 6: Batch Normalization model

Expectation- Until now the 4th model was better so I added batch normalization to the 4th model and constructed this model to work faster and more stable through normalization of the layer's inputs by re-centring and re-scaling so I expected this model to perform better than 4th model.

Result- The result surprised me again by making it faster it didn't perform better. In the 30 images it predicted 5 wrong images and also in Kaggle submissions it didn't perform well.

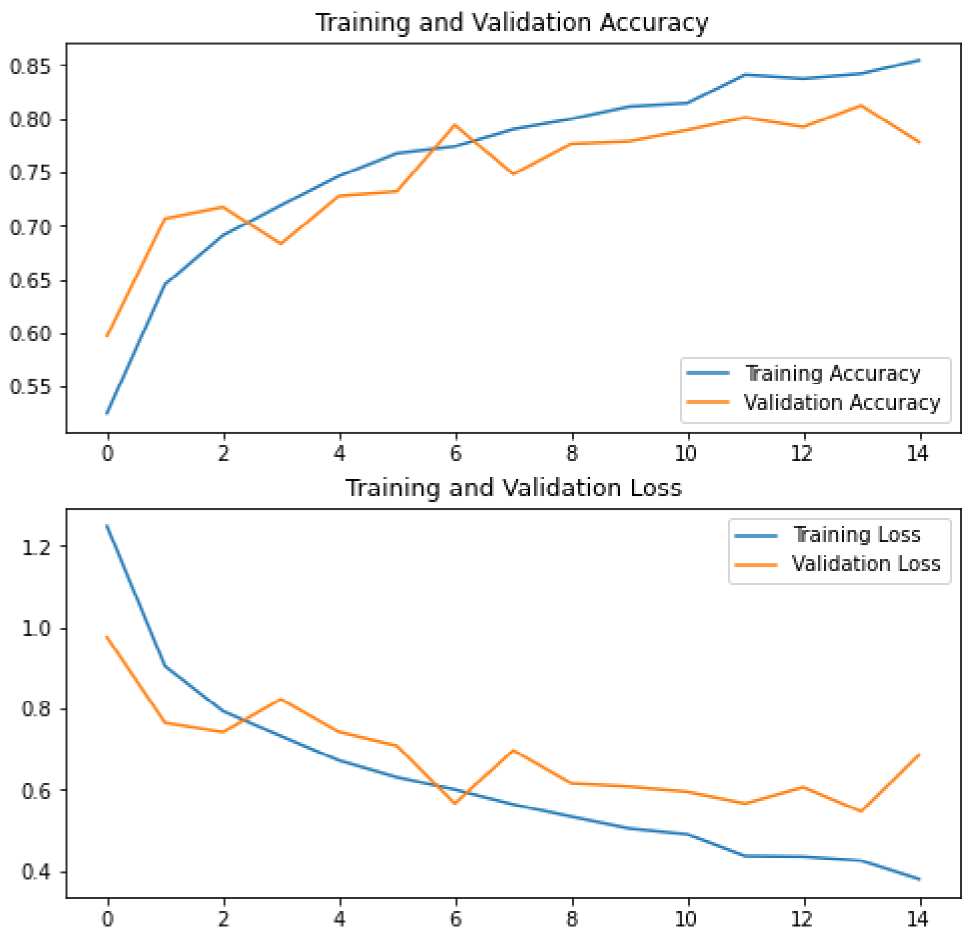
The total number of parameters for this model is

```
_________________________________________________________________
Total params: 339,174
Trainable params: 338,982
Non-trainable params: 192
_________________________________________________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 16s 44ms/step - loss: 0.3799 - accuracy: 0.8546 - val_loss: 0.6857 - val_accuracy: 0.7783
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



Now performing the models in different optimizers in the 4th model (best until now).

Model 7: Adagrad Optimizer model

Expectation- Using Adagrad we don't really need to tune the learning rate manually but here I have chosen the same learning rate to see the performance at the equal learning rate. Since the convergence is faster and more reliable, I expect it to perform better than the 4th model.

Result- The result was not what I was expecting rather it gave me eight wrong images in the 30 images

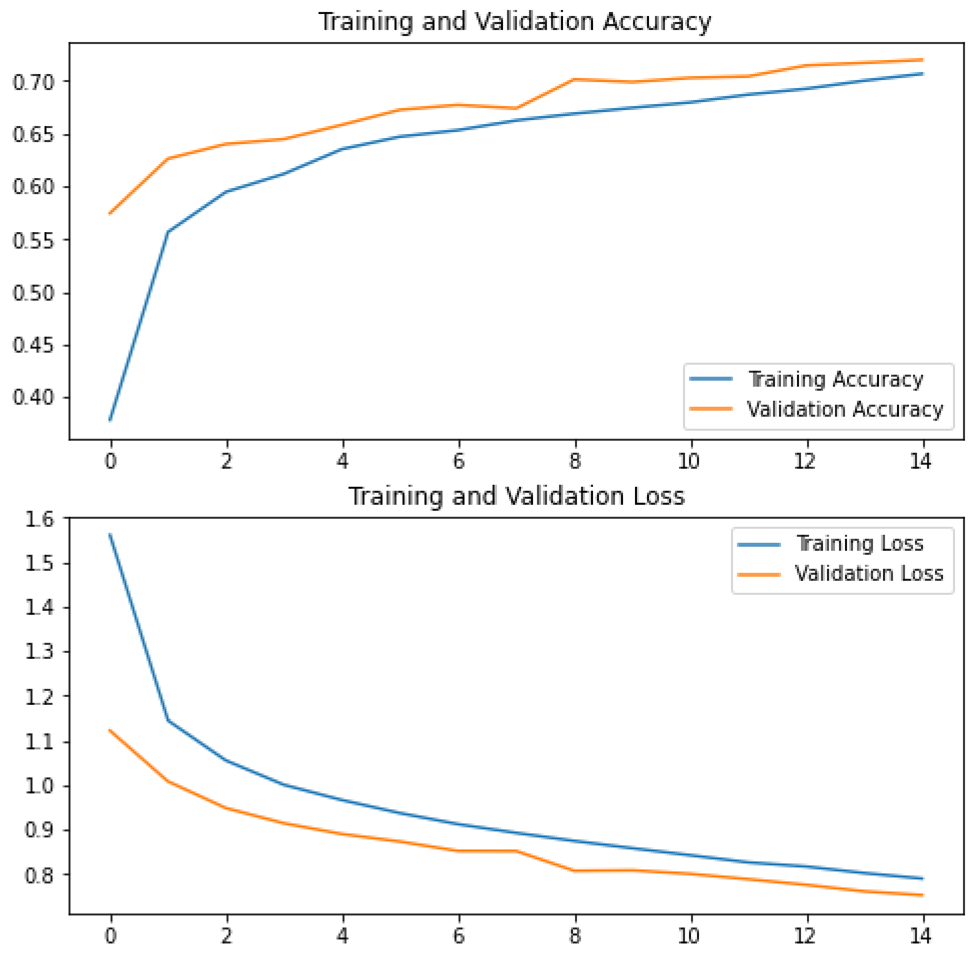
The total number of parameters for this model is

```
=================================================================
Total params: 338,790
Trainable params: 338,790
Non-trainable params: 0
_________________________________________________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 14s 40ms/step - loss: 0.7897 - accuracy: 0.7068 - val_loss: 0.7525 - val_accuracy: 0.7201
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



Model 8: Adadelta optimizer model

Expectation- Since each term is positive, this accumulated sum continues to grow throughout training, effectively shrinking the learning rate on each dimension so I expect that it would perform bad but I wanted to try it.

Result- The result came out to be as my expectation and even turning this specific model to be my worst model. It predicted 17 wrong predictions in the 30 images and hence resulted in my worst CNN model.

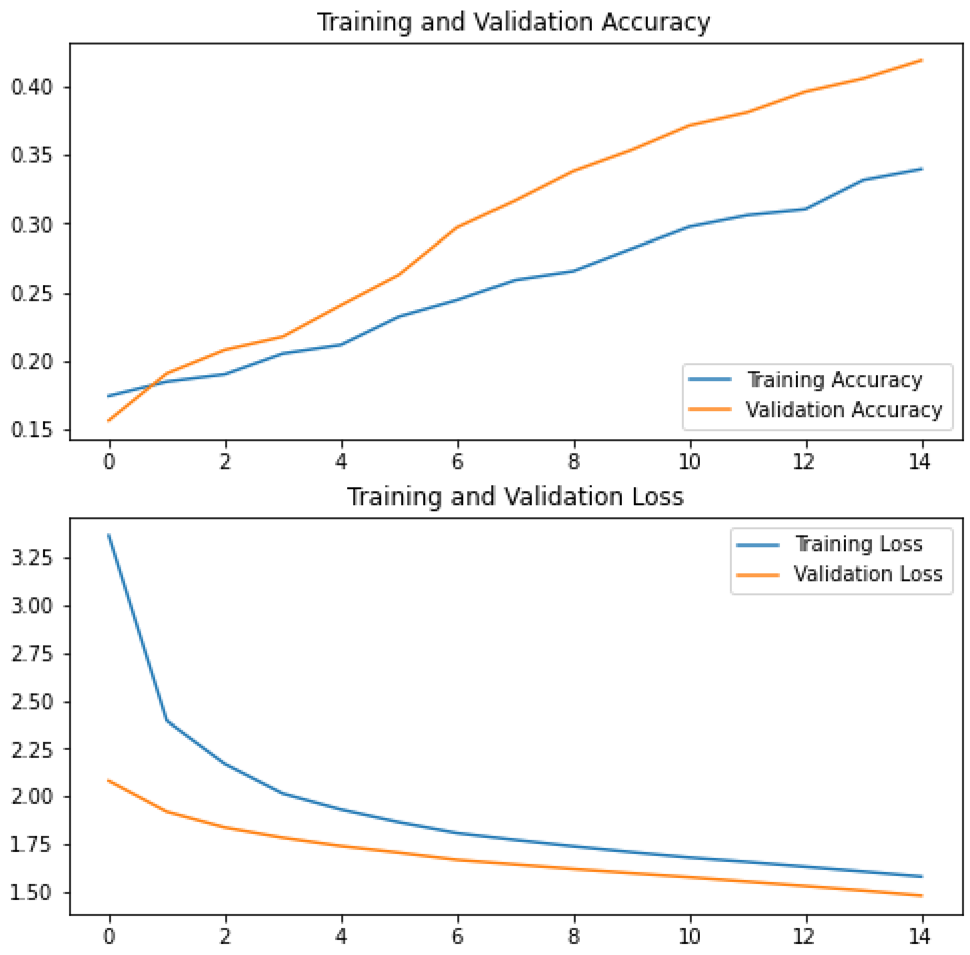
The total number of parameters for this model is

```
=================================================================
Total params: 338,790
Trainable params: 338,790
Non-trainable params: 0
_________________________________________________________________
```

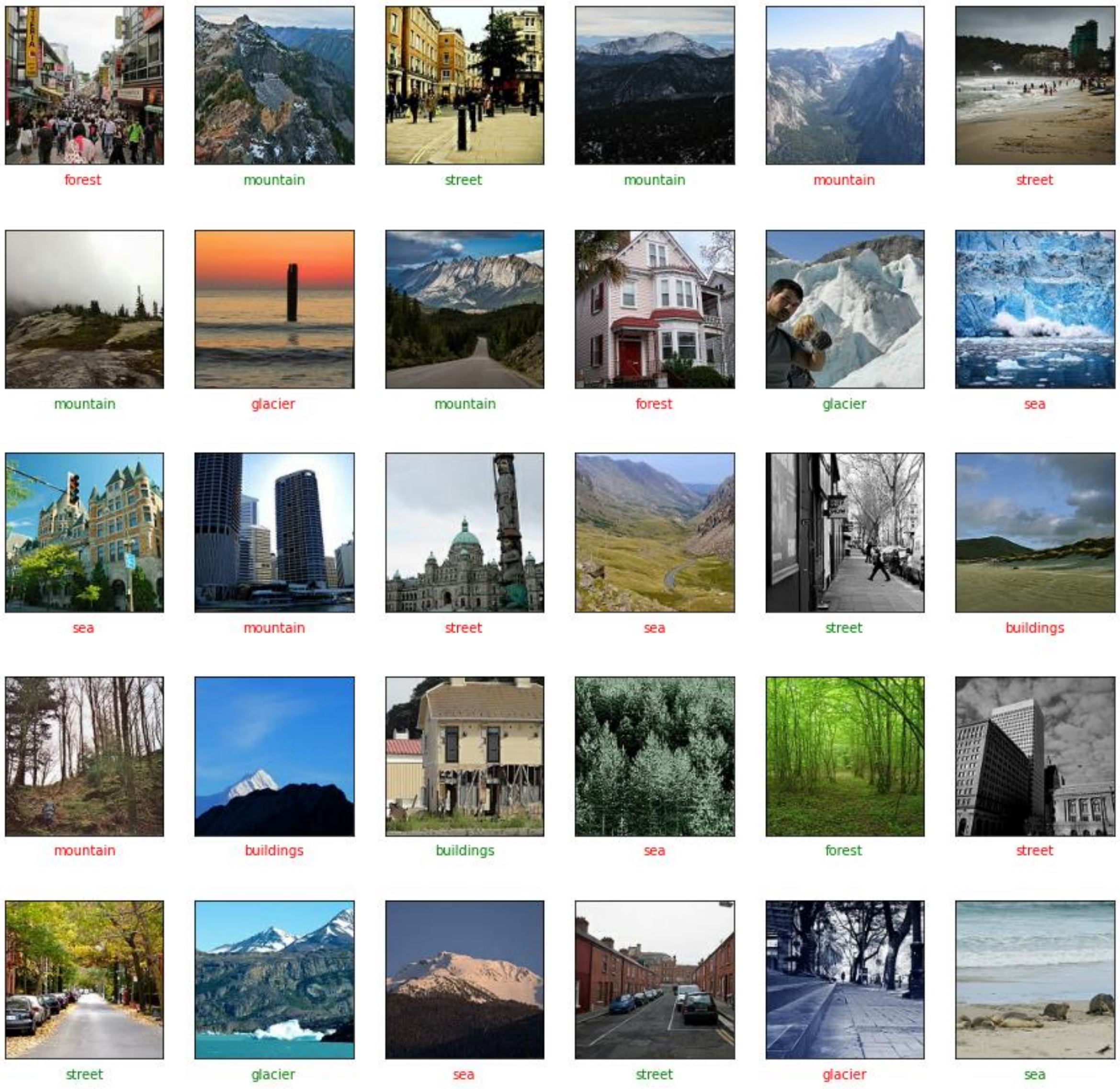
The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 15s 42ms/step - loss: 1.7275 - accuracy: 0.2874 - val_loss: 1.6440 - val_accuracy: 0.3457
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



The visual representation of how the model predicts the for 30 images can be seen below (worst model)



Model 9: Adamax Optimizer model

Expectation: This optimizer is a variant of Adam optimizer based on the infinity norm so I expect this model to perform similar to the 4th model.

Result- The result was same as my expectation since it is only a variant of Adam the results were quite similar in Kaggle prediction submission and also in the thirty images.

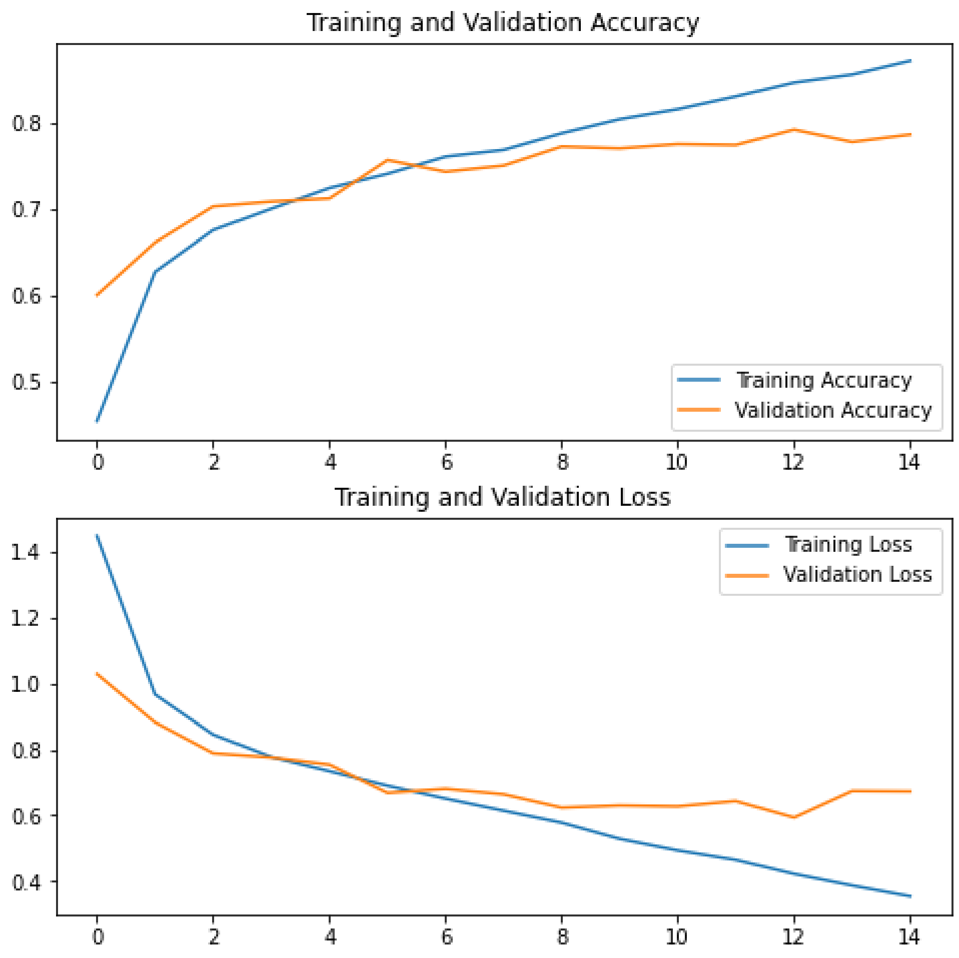
The total number of parameters for this model is

```
==========================================
Total params: 338,790
Trainable params: 338,790
Non-trainable params: 0
__________________________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 15s 44ms/step - loss: 0.3547 - accuracy: 0.8727 - val_loss: 0.6729 - val_accuracy: 0.7869
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



Model 10: More Batch Normalization model

Expectation- The reason why I am trying this is because the result in the previous batch normalization was not so bad so this time, I tried adding extra batch normalization to the model by batch normalizing almost every layers. So, I expect it to perform better than the previous batch normalization model.

Result- As I expected this gave me a better model than the previous batch normalised model but it was not the best model that is it didn't beat the 4th model in predicting.

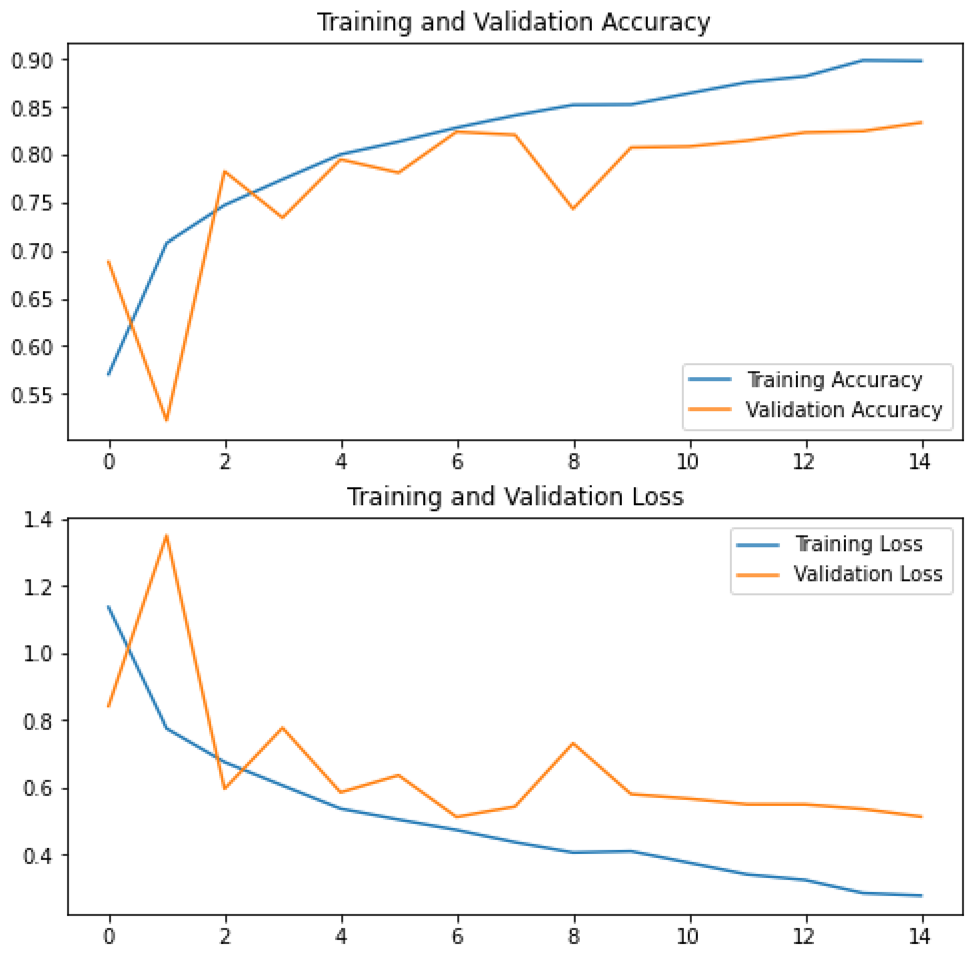
The total number of parameters for this model is

```
========================================
Total params: 358,950
Trainable params: 358,118
Non-trainable params: 832
________________________________________
```

The loss, accuracy, validation loss and validation accuracy at the last epoch was

```
Epoch 15/15
350/350 [==============================] - 20s 57ms/step - loss: 0.2768 - accuracy: 0.8984 - val_loss: 0.5127 - val_accuracy: 0.8338
```

The visual representation of the training and validation accuracy and also the training and validation loss can be seen in the picture



Model 11: More Dense layer in previous model

Expectation- My Kaggle leader board was not good until this model. In this model additional to batch normalization, I increased the number layers and also, I added additional dense layers at the end so this time I expect the model to perform better than all the models.

Result- The result satisfied me as I expected and it turned out to be the best model I constructed. Though it predicted more wrong images in 30 images than the previous best model that is the 4th model but in Kaggle submission it gave a lot better result.

The results were shown above

Comparison table:

Model	Name	Padding	Striding	Loss	Accuracy	Val_loss	Val_acc
0	Original	No	No	0.3309	0.8842	0.8856	0.7701
1	Additional Layer	No	No	0.4339	0.8392	0.5724	0.8155
2	Removing Layer	Yes	No	0.1701	0.9552	1.9646	0.6468
3	Random Layer	Yes	Yes	0.0917	0.9723	1.7786	0.6868
4	Adjusted Layer	Yes	Yes	0.2640	0.9021	0.9201	0.7733
5	Additional Layer (4th)	Yes	Yes	0.6259	0.7630	0.6340	0.7580
6	Batch Norm	Yes	Yes	0.3799	0.8546	0.6857	0.7783
7	Adagrad	Yes	Yes	0.7897	0.7068	0.7525	0.7201
8	Adadelta	Yes	Yes	1.725	0.2874	1.6440	0.3457
9	Adamax	Yes	Yes	0.3547	0.8727	0.6729	0.7869
10	More Batch Norm	Yes	Yes	0.2768	0.8984	0.5127	0.8338
11	Best Model	Yes	Yes	0.3973	0.8583	0.4317	0.8516

Conclusion:

I have performed other models with various number of filter size in the best models but the result as not best. The best model had a greater number of layers and almost every layer was normalised using the batch normalisation. But I prefer a CNN with a smaller number of layers but high performance because the time consumption increases as the layers increase in the CNN. And also, the one with the batch normalised CNN model there was a difference than the rest of the models in the non-trainable parameters. It was 0 for every model except the one which uses batch normalisation.

Reaction and reflection:

Overall, the assignment was more fun than the other assignments which I had done in this course because it was not more challenging as the other assignments rather it was a little time consuming than the other assignments. But I had really enjoyed doing this assignment and learnt a lot about CNNs hyperparameter tuning.