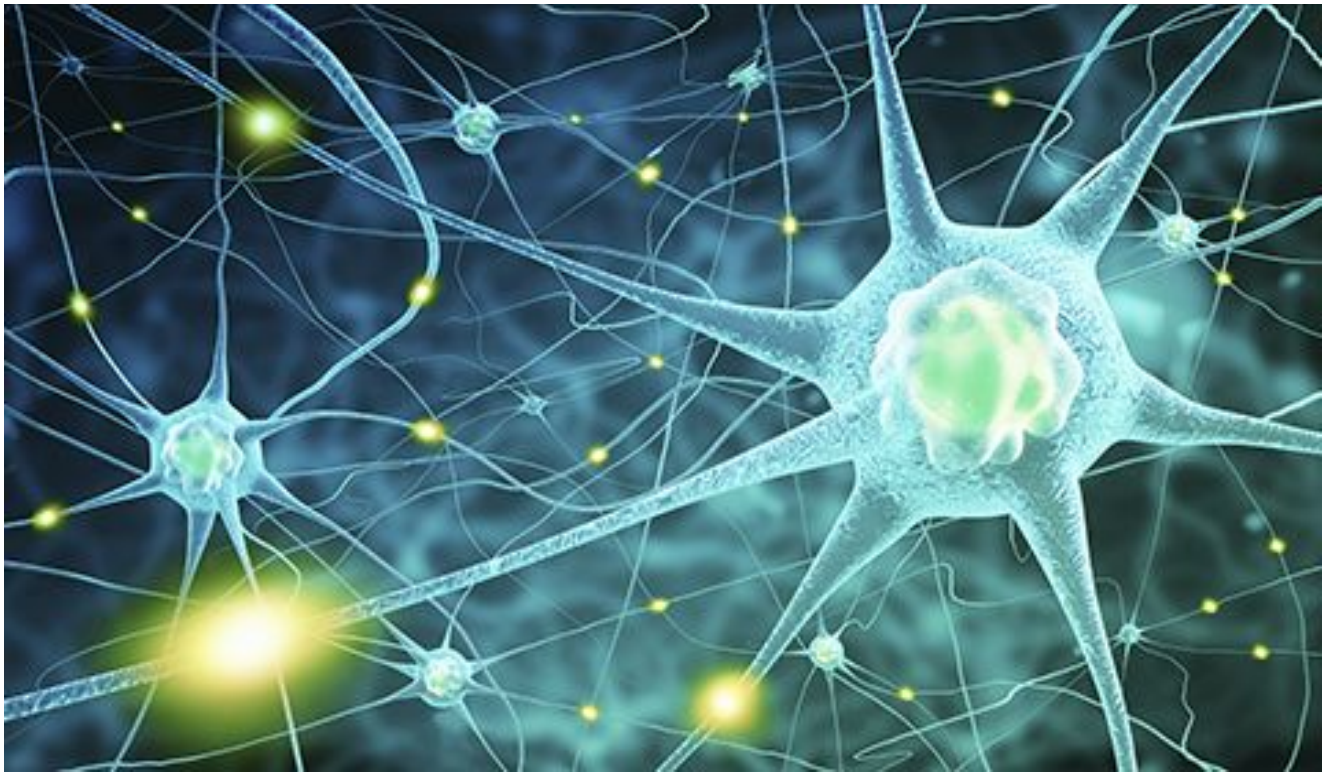
Assignment 4

Deep Learning - CS577



Akshay R

A20442409

Task 1:

Problem Statement:

To perform binary classification on a modified cats and dogs dataset using different CNN models. Implementing a model with pre trained weights (VGG16) with and without augmentation and fine tuning the model

Proposed Solution:

Kaggle's Cats and Dogs dataset contains 25000 images of cats and dogs. To simulate the condition of limited data 4000 images are chosen, 2000 cats and 2000 dogs. This is divided into test, train and validation datasets.

First a CNN model is created and evaluated. Then a new model is created with VGG16 as the base model, with pre-trained weights. This model is trained while freezing the weights of the VGG16 model. This model is fine tuned by training the model while unfreezing the top layers of the VGG16 model. Data augmentation is performed on training data and a new model is trained on augmented images.

Implementation ,methodology:

Kaggle's dataset is divided into train and test datasets with a 80% -20% split. The train dataset is further divided into 80%- 20% validation splits. The final number of images are as follows:

Train: Cats: 1280 images and Dogs: 1280 images

Validation: Cats: 320 images and Dogs: 320 images

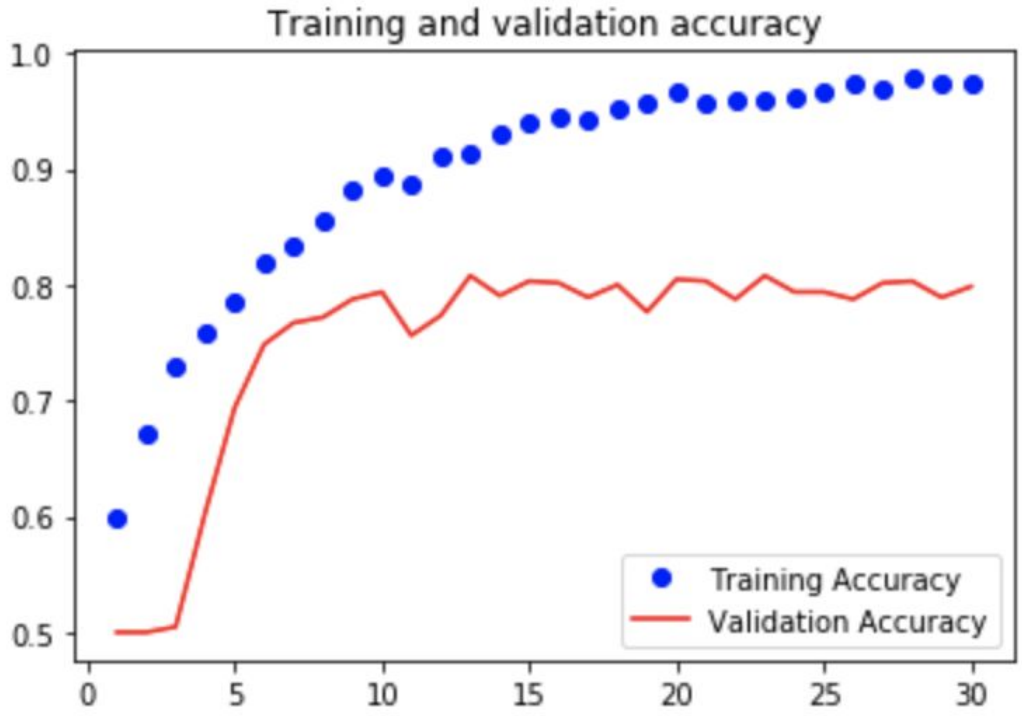
Test: Cats: 400 images and Dogs: 400 images

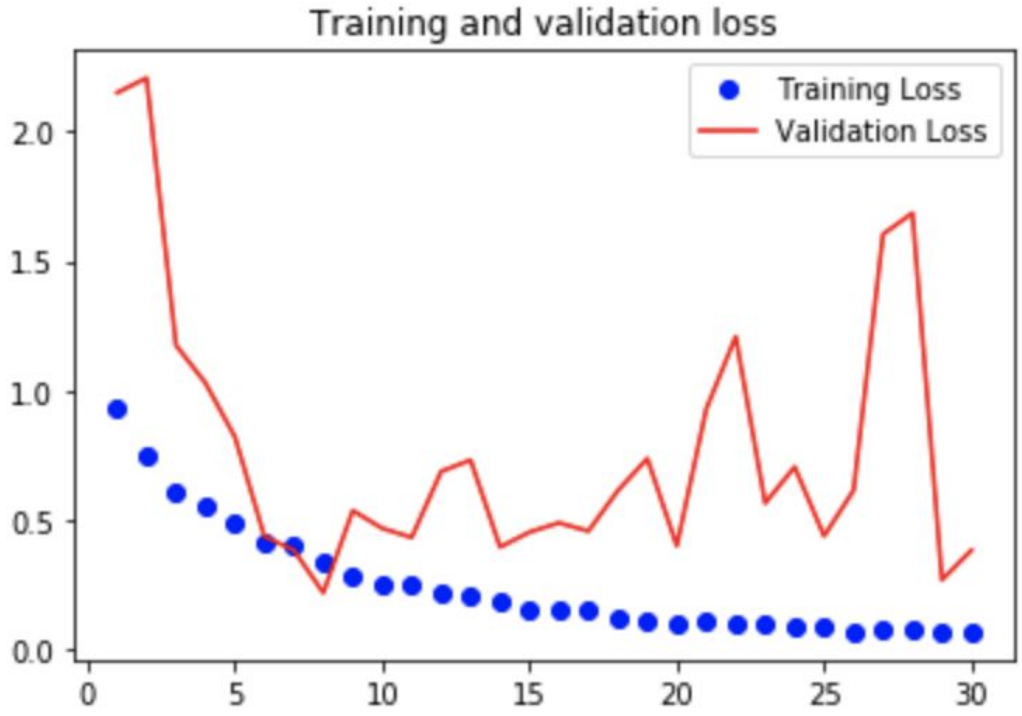
This data is passed through keras datagenerator, flow from directory function to load the images, normalize them into train, validation and test generators.

A keras sequential model is created with the following architecture:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 74, 74, 32)	0
batch_normalization_1 (Batch	(None, 74, 74, 32)	128
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 36, 36, 64)	0
batch_normalization_2 (Batch	(None, 36, 36, 64)	256
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 17, 17, 128)	0
batch_normalization_3 (Batch	(None, 17, 17, 128)	512
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 128)	0
batch_normalization_4 (Batch	(None, 7, 7, 128)	512
dropout_1 (Dropout)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
batch_normalization_5 (Batch	(None, 512)	2048
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1)	513

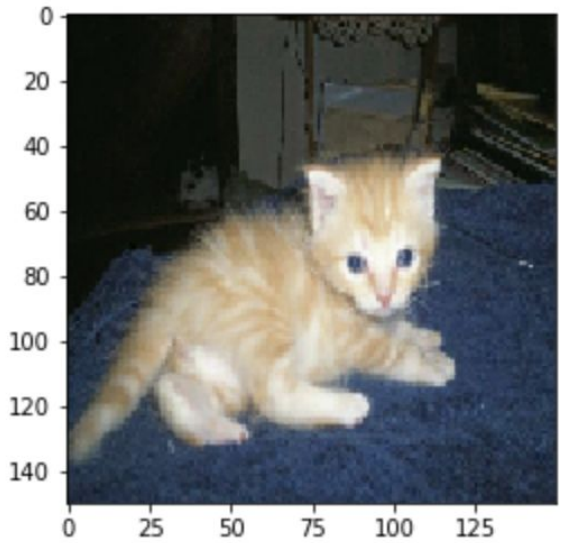
This model is trained on train datagen images with RMSprop as the optimizer with learning rate of 1e-5. The loss function used is Binary crossentropy. A validation accuracy of 80% is achieved. The plot of validation loss vs training loss and validation accuracy vs training accuracy is as shown below:





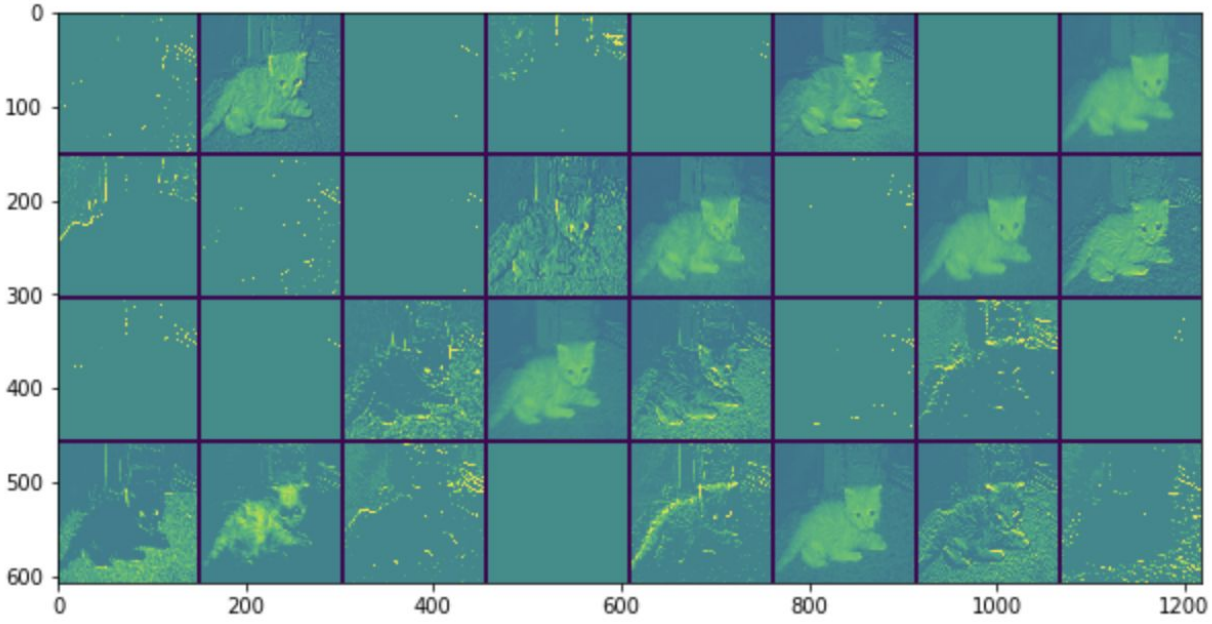
It is seen that the model starts to overfit around the 8th epoch and hence training must be stopped here.

The activations of the model are visualised on a specific image shown below.

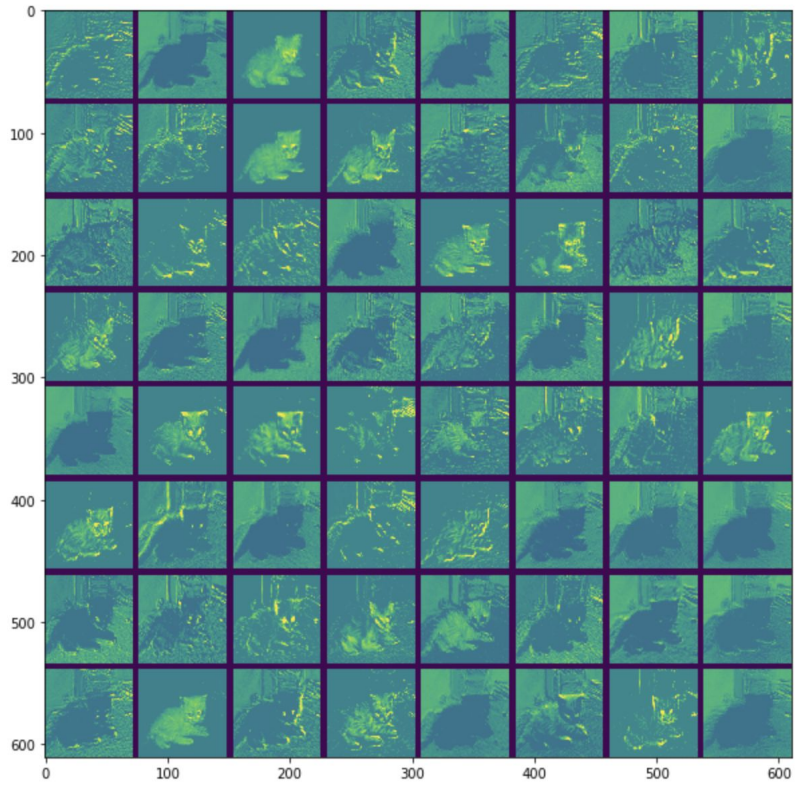


The layer activations observed in the first three activation layers are as shown below:

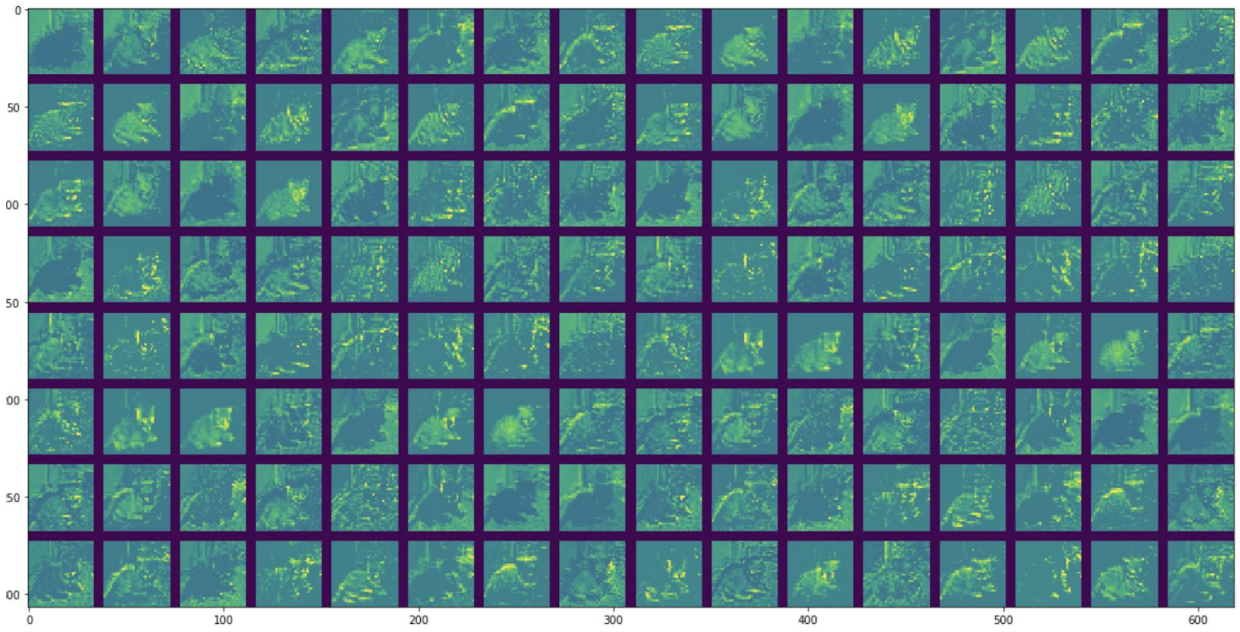
Conv2d_1:



Conv2d_2:



Conv2d_3:

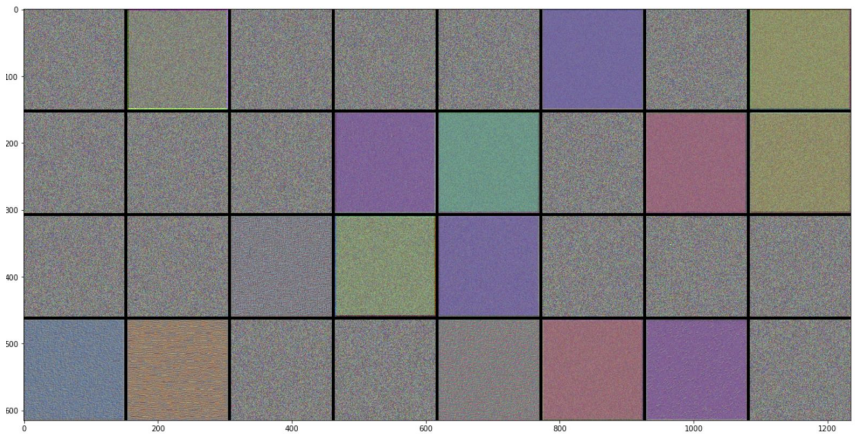


It is noticed that the initial layers performs some basic edge detection and the deeper layers perform much detailed feature extraction. Some activations are not activated as

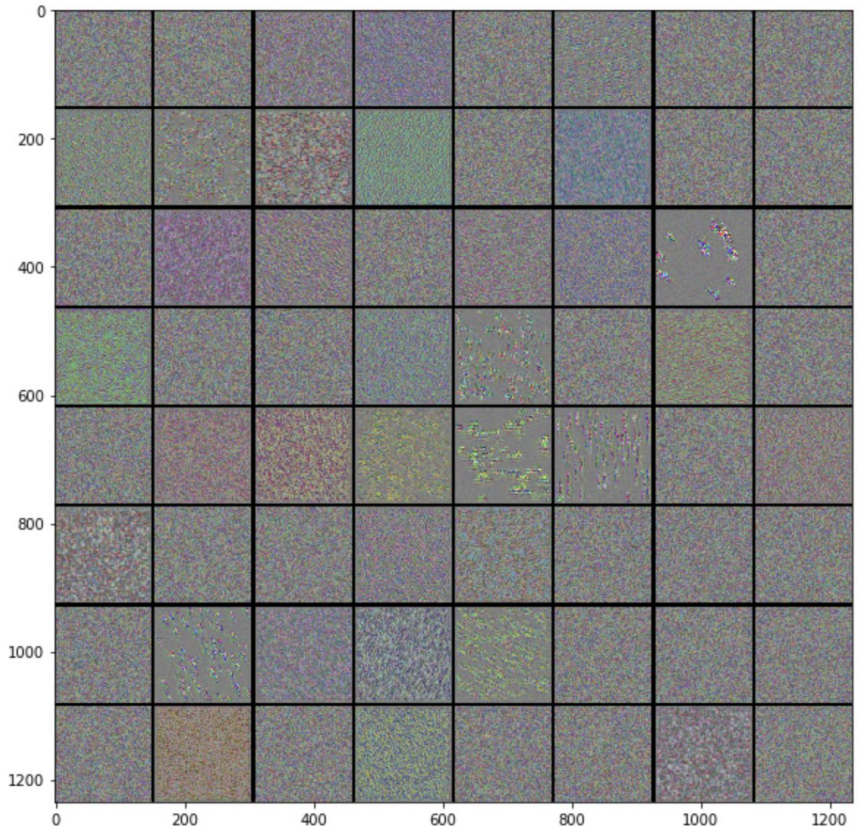
they might belong to a feature that is not present in the current image.

Next filter weights are visualized by calculating the gradients with respect to a random generated input. This is plotted and the results are as below:

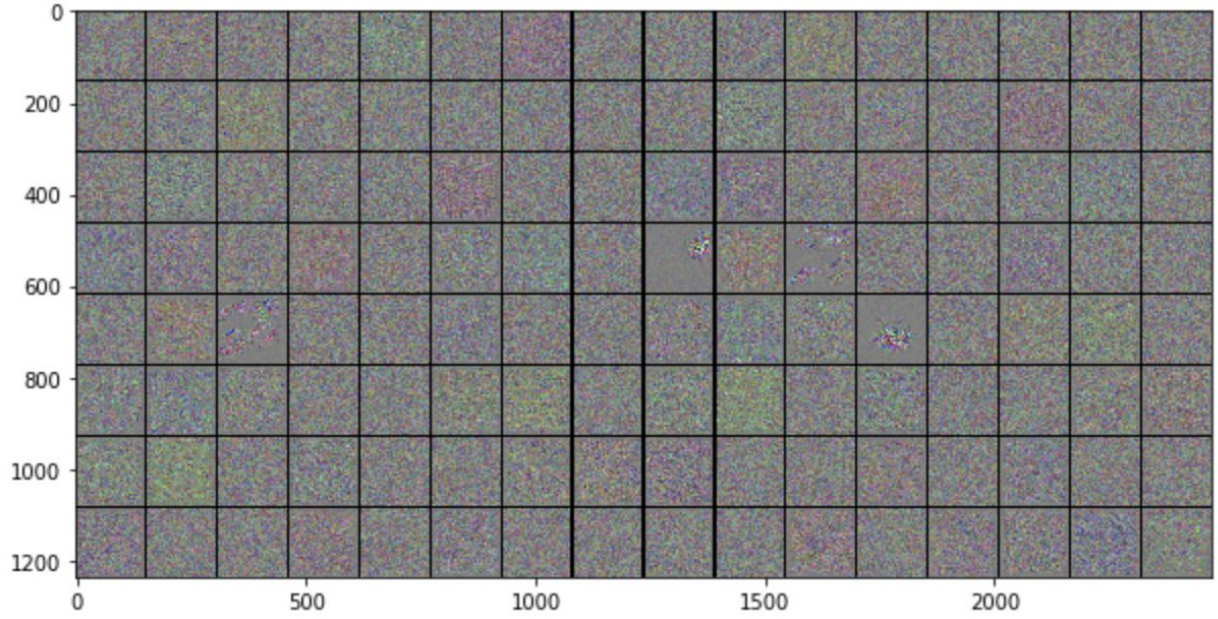
Conv2d_1:



Conv2d_2:



Conv2d_3:

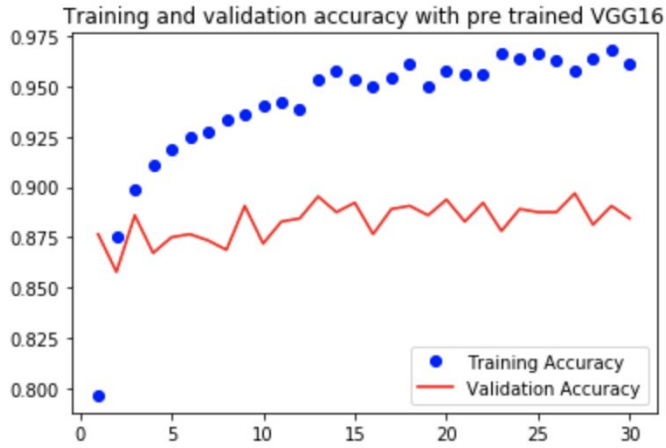


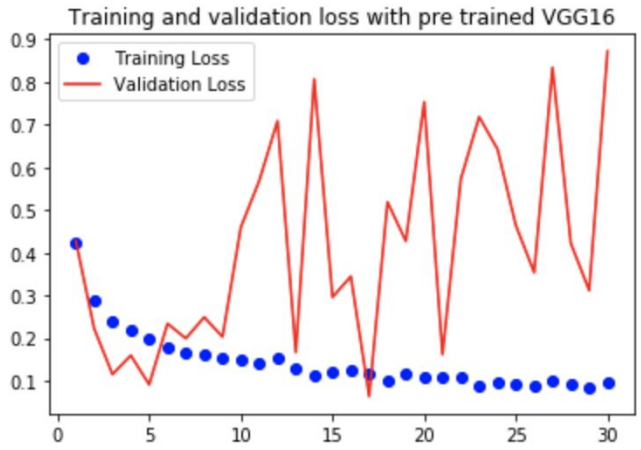
It is observed that the initial layer performs some kind of color detection and some basic pattern recognition. Deeper layers perform some kind of texture recognition, much more complex than the initial layers.

Next a new model is created with VGG16 as the base model. The weights of the base network are frozen. The model architecture are as shown below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 8192)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 8192)              0
_________________________________________________________________
dense_3 (Dense)              (None, 356)               2916708
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 357
=================================================================
Total params: 17,631,753
Trainable params: 2,917,065
Non-trainable params: 14,714,688
_________________________________________________________________
```

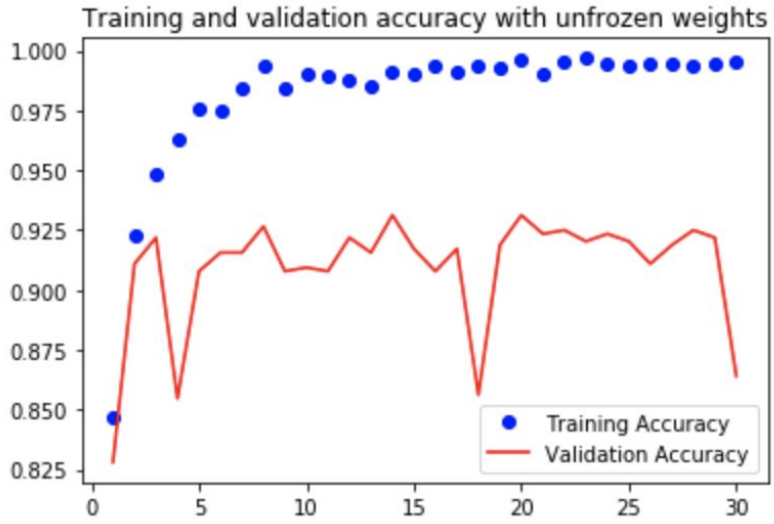
This model is trained on the train datagen and a validation accuracy of 89% is achieved. The plot of validation loss vs training loss and validation accuracy vs training accuracy is as shown below:

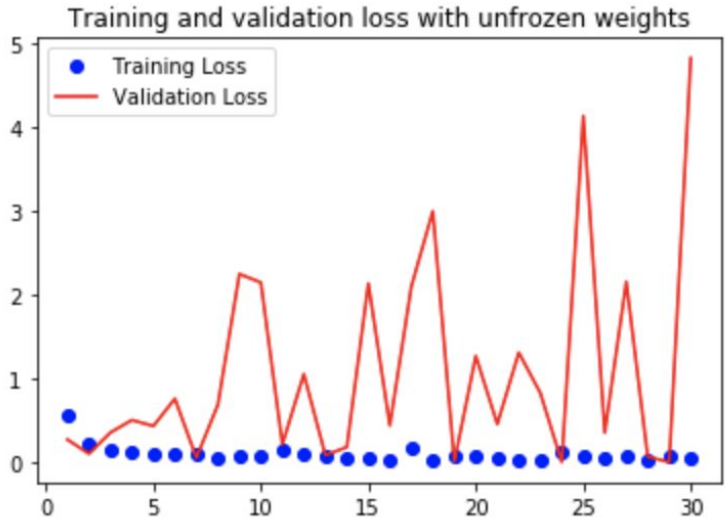




It is seen that this model quickly starts to overfit due to the addition of pre-trained networks. The training must be stopped at 5 epochs.

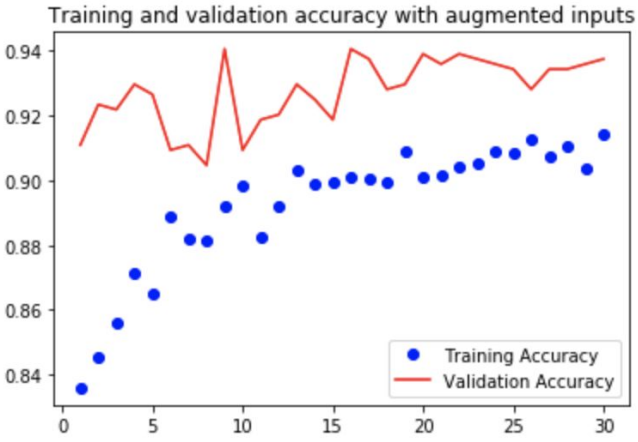
Nwo the top layer's weights are unfrozen and the model is trained again on the image data. This results in fine tuning the model to meet the requirements of our specific problem. The model generates an output validation accuracy of 93%. The plot of validation loss vs training loss and validation accuracy vs training accuracy is as shown below:

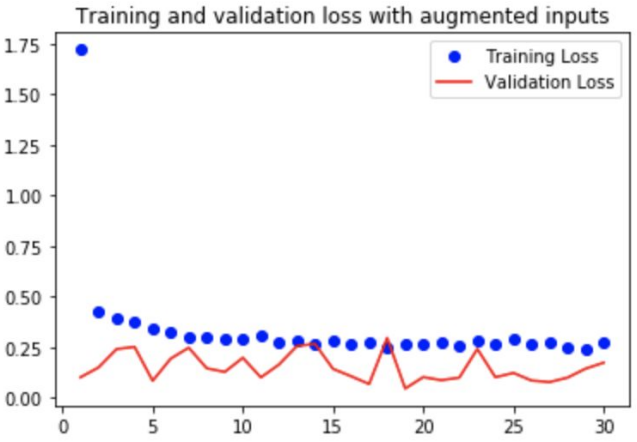




This is the best model achieved so far with just 2000 images. We finally create out final model similar to the model with VGG16 as the base network but is trained on a much larger dataset containing augmented images.

The model is exactly the same as the one above. This model generates a validation accuracy of 94%. The plot of validation loss vs training loss and validation accuracy vs training accuracy is as shown below:





Results and discussion:

After observing the result of the model designed above it is noted that the performance of the model improved with using a pre trained network. Fine tuning this model to our problem statement increased the performance.

Data augmentation overcame the problem of having less number of images to work with and finally achieved an accuracy of 94%.

Task 2:

Problem Statement:

Multi-Class classification of images from CIFAR10 dataset using CNNs with inception and residual blocks.

Proposed Solution:

The CIFAR10 dataset has 60,000 images,60,000 labels distributed over 10 classes, 11,000 images and labels in each class. The data obtained must be normalized and further divided into train, validation and test data sets. The labels must be encoded into a categorical variable with three values.

A CNN model is created with several Convolution, MaxPooling and Batch Normalization layers and is trained on the training sample. An inception block is added to this model and the results are compared. Finally the inception model is removed and a residual network is added and the results are compared.

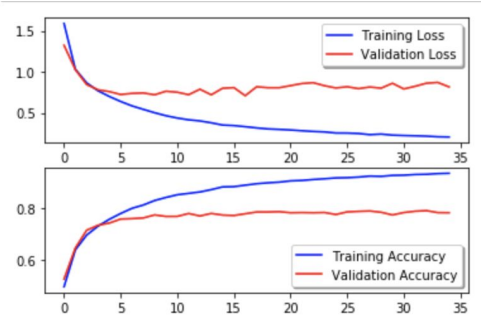
Implementation and methodology:

The given dataset is first divided into test and train data.The data is standardized by subtracting the mean and dividing the standard deviation.

A CNN model is created with the architecture as shown below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 128)       36992
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               4194816
_________________________________________________________________
batch_normalization_3 (Batch (None, 512)               2048
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 4,240,522
Trainable params: 4,239,178
Non-trainable params: 1,344
```

This model is trained on the training images. The loss function used is Categorical crossentropy with RMSprop as the optimizer. The results are plotted and are as show below:

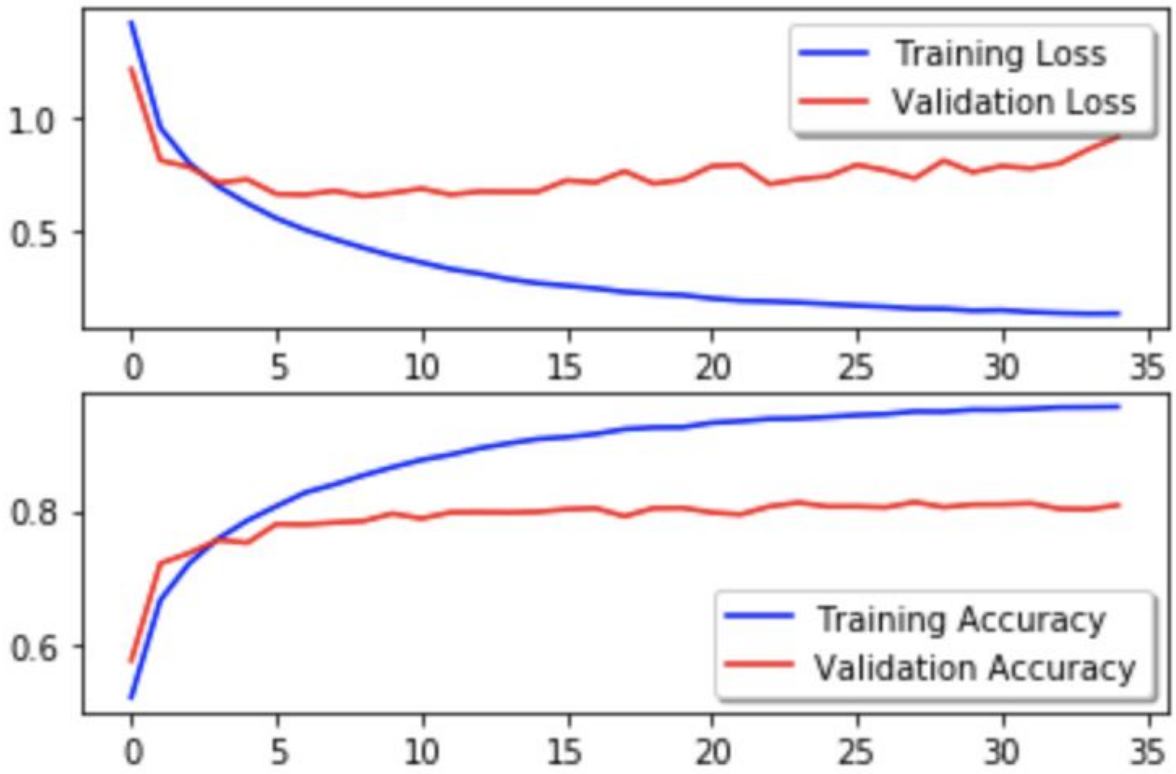


The stats are as follows:

```
Validation: accuracy = 0.781000  ;  loss_v = 0.816508
Test: accuracy = 0.784700  ;  loss = 0.818066
```

The results obtained are stored and a new model is created. In the new model a inception block is added to more complex feature extraction with a deeper network. The inception block allows us to go deeper with substantially less number of parameters if there vere just more convolution blocks. This is because of the architecture of the inception block.

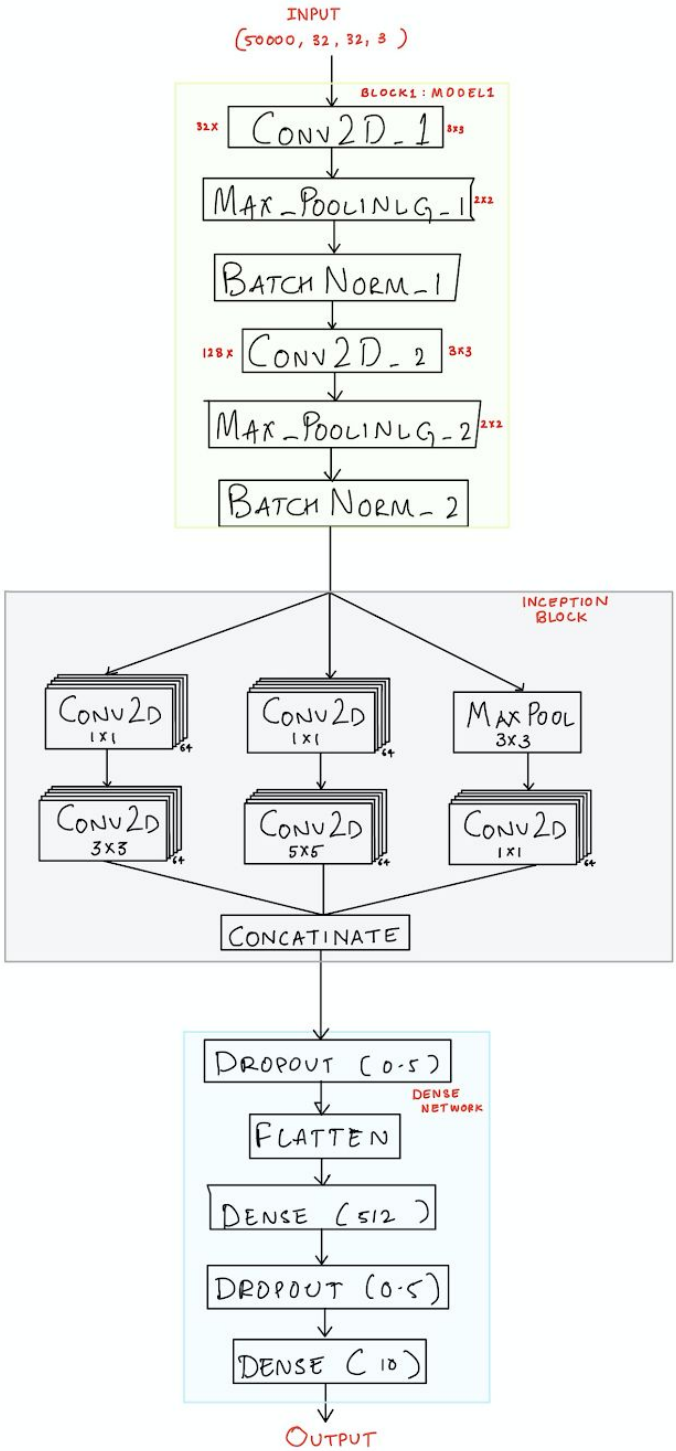
The picture in the next page displays the architecture of the model used. This model is trained on the training images and the results are evaluated. The performance of the model over 35 epochs are displayed below:



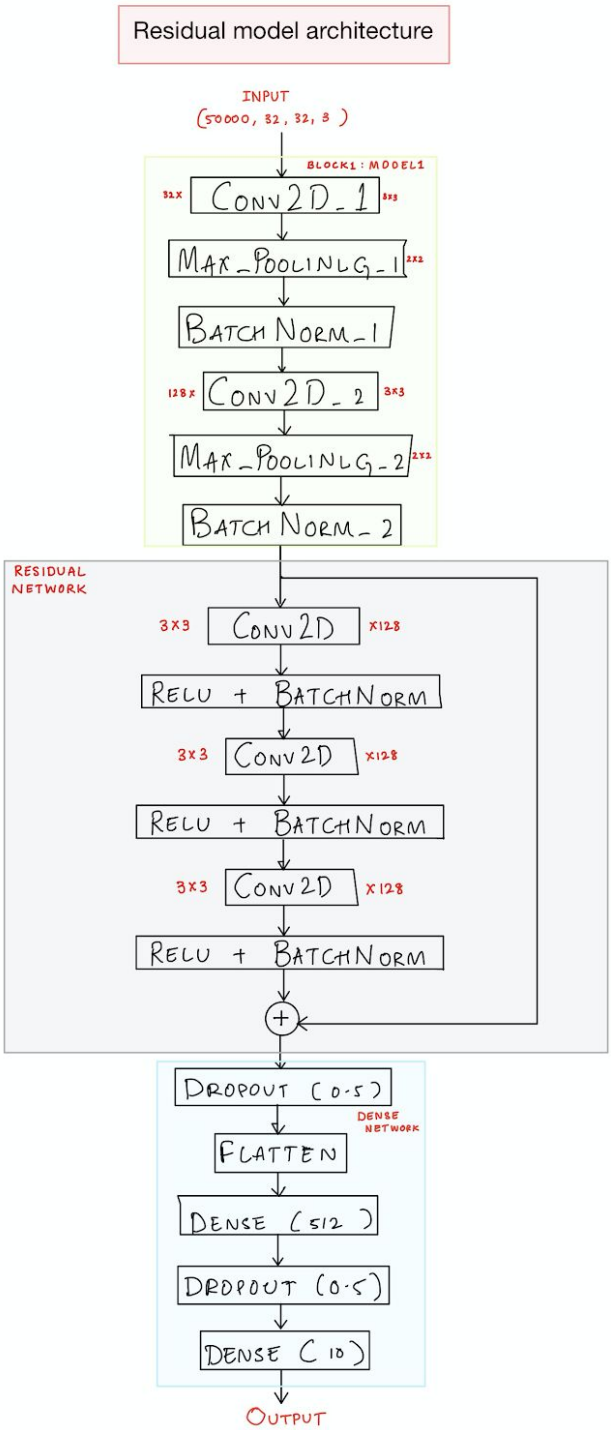
This model generated the following results:

```
Validation: accuracy = 0.808100  ;  loss_v = 0.912779
Test: accuracy = 0.804900  ;  loss = 0.910464
```

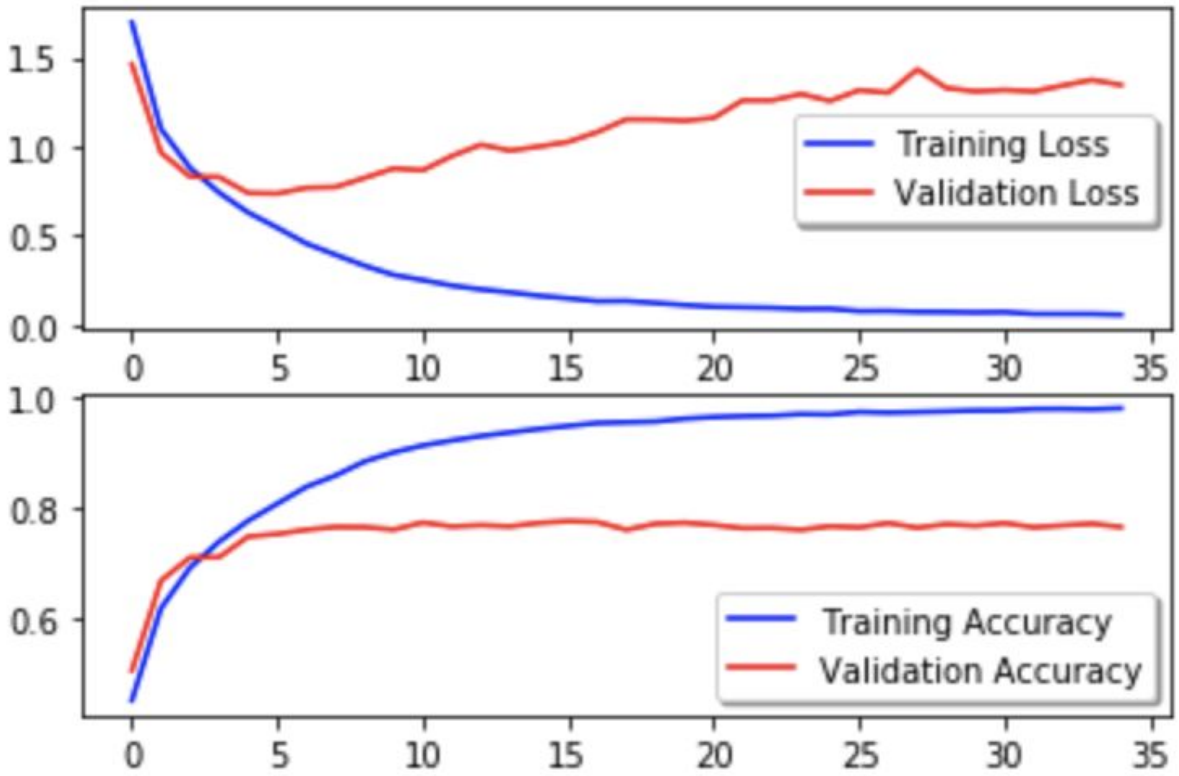
Inception model architecture



For the next model we remove the inception block, add a residual block and evaluate its performance. The architecture of this model is as follows:



This model is trained on the training images and the results are evaluated. The performance of the model over 35 epochs are displayed below:



The results are as follows:

```
Validation: accuracy = 0.766500  ;  loss_v = 1.344585
Test: accuracy = 0.764400  ;  loss = 1.371122
```

Results and discussion:

After observing the result of the model designed above it is noted that adding an inception block increased the accuracy of the model to 81%. However the addition of a residual block decreased the performance of the model.

The increase in the performance is justified by using more layers for better feature extraction. The decrease in the performance of the model can be justified by saying that a combination of more residual blocks might be required for better feature extraction.