CS 577

HW #4

Khalid Saifullah

ID : A20423546
Semester : Spring 2021
Date : 27th April 2021

Answer to the Questions

① Original Image : 1 4×4 RGB image

R-channel

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

G-channel

2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

B-channel

1	1	1	1
2	2	3	2
3	3	3	3
4	4	4	4

filter

1	1	1
1	1	1
1	1	1

After convolution :

R-channel

9	9
9	9

+

G-channel

18	18
18	18

+

B-channel

18	18
27	27

⇒

Output

45	45
54	54

② After convolution with zero padding :

R-channel

4	6	6	4
6	9	9	6
6	9	9	6
4	6	6	4

+

G-channel

8	12	12	8
12	18	18	12
12	18	18	12
8	12	12	8

+

B-channel

6	9	9	6
12	18	18	12
18	27	27	18
14	21	21	14

⇒

output

18	27	27	18
30	45	45	30
36	54	54	36
26	39	39	26

③ Atrous convolution with a dilation rate of 2 (having zero padding)

R-channel

4	4
4	4

+

G-channel

8	8
8	8

+

B-channel

12	12
8	8

⇒

Output

24	24
20	20

④ When an image is convolved with a filter, the value that will be assigned to the output ~~of~~ pixel depends on how well the filter matches that patch of the image. If the filter matches very well with the patch of the image, then the output pixel will have a high value ~~Thus convolution as~~ because when we do the convolution, we multiply the filter with the image. Thus, convolution can be interpreted as template matching.

⑤ As we are pooling between layers (or doing convolution with a stride greater than 1), we are sampling the spatial dimensions and hence we get an image pyramid with different spatial resolutions at different layers. Therefore, a fixed size convolution ~~covers~~ filter covers a bigger spatial region in the upper layers of the pyramid.

⑥ When convolution is performed on an image with a filter, there is loss of information (coefficients) in the output layer. As spatial dimensions decrease, depth can be increased (using multiple ~~bayes~~ filters layer by layer) to compensate for the reduced coefficients (i.e. keep the same number of coefficients).

⑦ Size of the resulting tensor $= 126 \times 126 \times 16$

⑧ Produced size $= \frac{w - h + 2p}{s} + 1$

$= \frac{128 - 3 + 0}{2} + 1$

$= 63.5$

$\simeq 63$

So, output tensor size $= 63 \times 63 \times 16$

(9) Using 1×1 convolution, we can reduce the number of channels to our desired value.

Let's consider,

input image size = $128 \times 128 \times \underline{32}$

Using 8 filters of size = 1×1

output tensor is reduced to $128 \times 128 \times \underline{8}$

no. of channels reduced

(10) The convolution layers are used to extract image pattern or template. With deeper convolutional layers, the filters learn more sophisticated details of the image. For instance, the early layers may learn basic features like edges and lines, whereas the deeper layers' filters may learn more complex features, like eyes, nose, objects etc.

(11) Using max-pooling on image I (4×4×3) with a 2×2 filter with a stride of 2,

R-channel		G-channel		B-channel	
1	1	2	2	2	2
1	1	2	2	4	4

(12) The purpose of pooling is to downsample the spatial dimension of the image without changing the depth, reducing the number of coefficients. However, there is no learning in this step.

(13) The purpose of data augmentation is to introduce more variability in the training data and generalization of the networks. It is useful in a situation when the amount of training data is less and the network tends to overfit.

(14) The purpose of transfer learning is to improve learning in the target task by leveraging knowledge from the source task. It is useful when there is not sufficient data to perform training and the target data is similar to the source data.

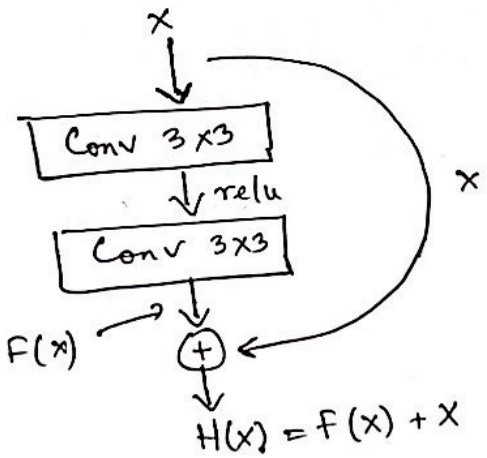
(15) Transfer learning involves the use of a pre-trained network to help him train a new network. In this method, usually the pretrained network is used to extract the low-level features quickly. So, if the weights of the pretrained network is not frozen, then the network starts learning, and destroys the purpose of the pre-trained weights. This is why we generally freeze the coefficients of a pre-trained network.

(16) Initially, the coefficients of the ~~t~~ pretrained network is frozen while the untrained layers go through training. Then the layers of the pre-trained network can be unfrozen for fine tuning of the coefficients. We can also unfreeze specific layers of the pretrained network.

(17) It's a combination of filters with multiple sizes operating on the same level and ~~the~~ each output concatenated into a single output tensor forming the input of the next stage. This can detect different size variation in the location of the information ~~and~~ as it

allows the network learn the best weights ~~when~~ and automatically select the more useful features.

(18) The advantage of residual block is that it is easier to learn $F(x)$ residual compared with $H(x)$ (i.e. learn deviation from identity instead of function). Besides, the skip connection also eliminates the problem of vanishing gradients.



Programming Part

1. Cats and Dogs

(a) Firstly, the Cats and Dogs dataset was downloaded from [1]. From the compressed file, 2000 pictures of cats, and 2000 pictures of dogs are extracted. Two subdirectories, [‘train’, ‘test’], are created under the parent directory ‘Dataset’. Two additional folders, namely [‘cats’, ‘dogs’], are created inside each of the train and test subdirectory. All of this is done so that the Keras Image Data Generator class and flow_from_directory() API could be used. 1500 cat and 1500 dog images (3000 images in total) are used for training, and 500 cat and 500 dog images (1000 images in total) are used for validation. Test is done on 2000 additional images (1000 cats and 1000 dogs).

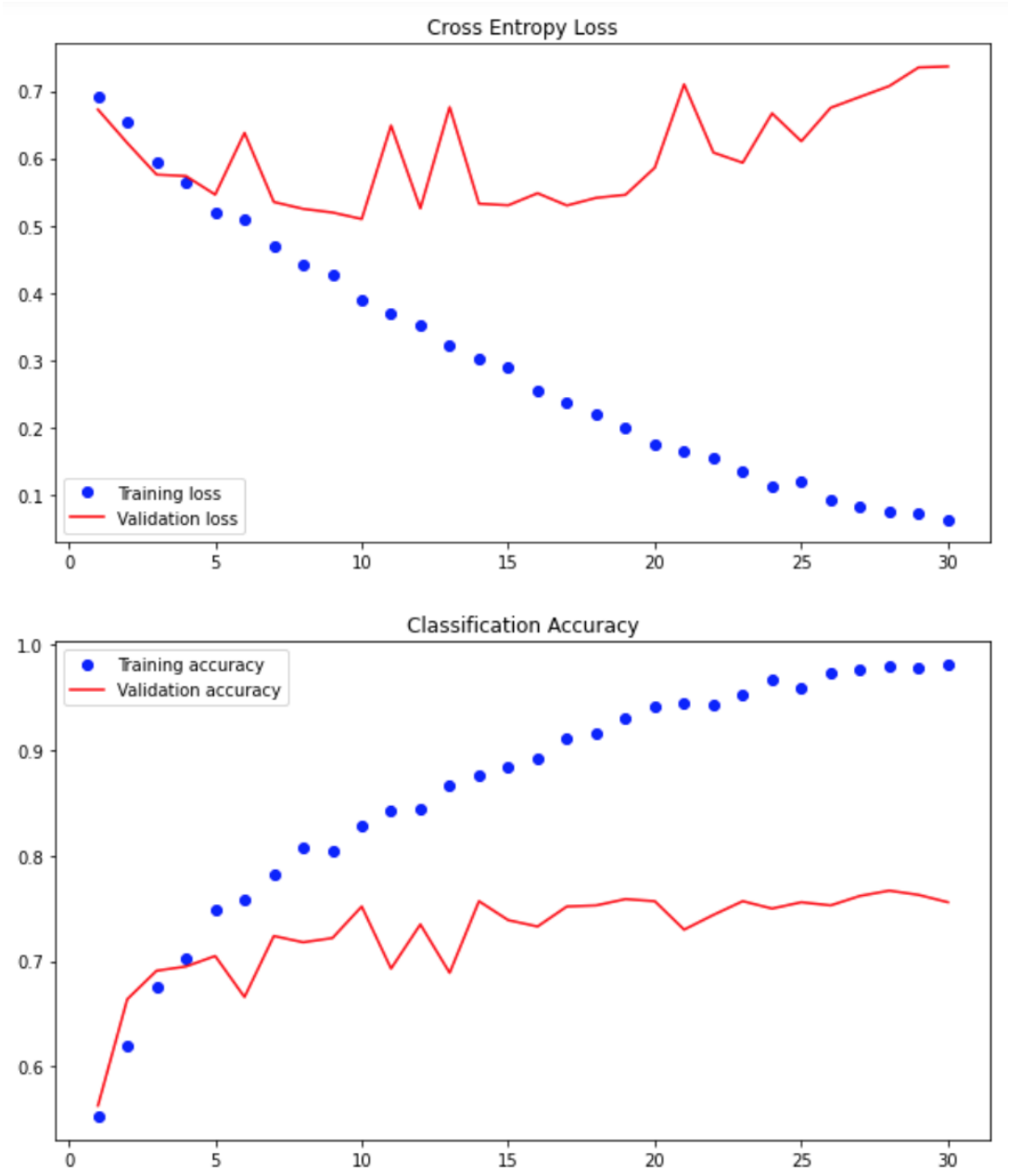
(b) Training, validation, and testing data generators are defined with a batch size of 60, and target size of (200, 200).

(c) In this part, a convolutional neural network has been defined and the model summary is shown below.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 200, 200, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 100, 100, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 98, 98, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 47, 47, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 23, 23, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 12800)             0
_________________________________________________________________
dense (Dense)                (None, 512)               6554112
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 6,795,457
Trainable params: 6,795,457
Non-trainable params: 0
```

Relu activation function has been used for the convolutional layers and also for the first fully connected dense layer after flattening. And sigmoid activation function has been used for the last fully connected layer since it is a binary classification problem. The model parameters are:
Optimizer – RMSprop (lr = 1e-4)
Loss – Binary cross entropy
Metrics – Accuracy

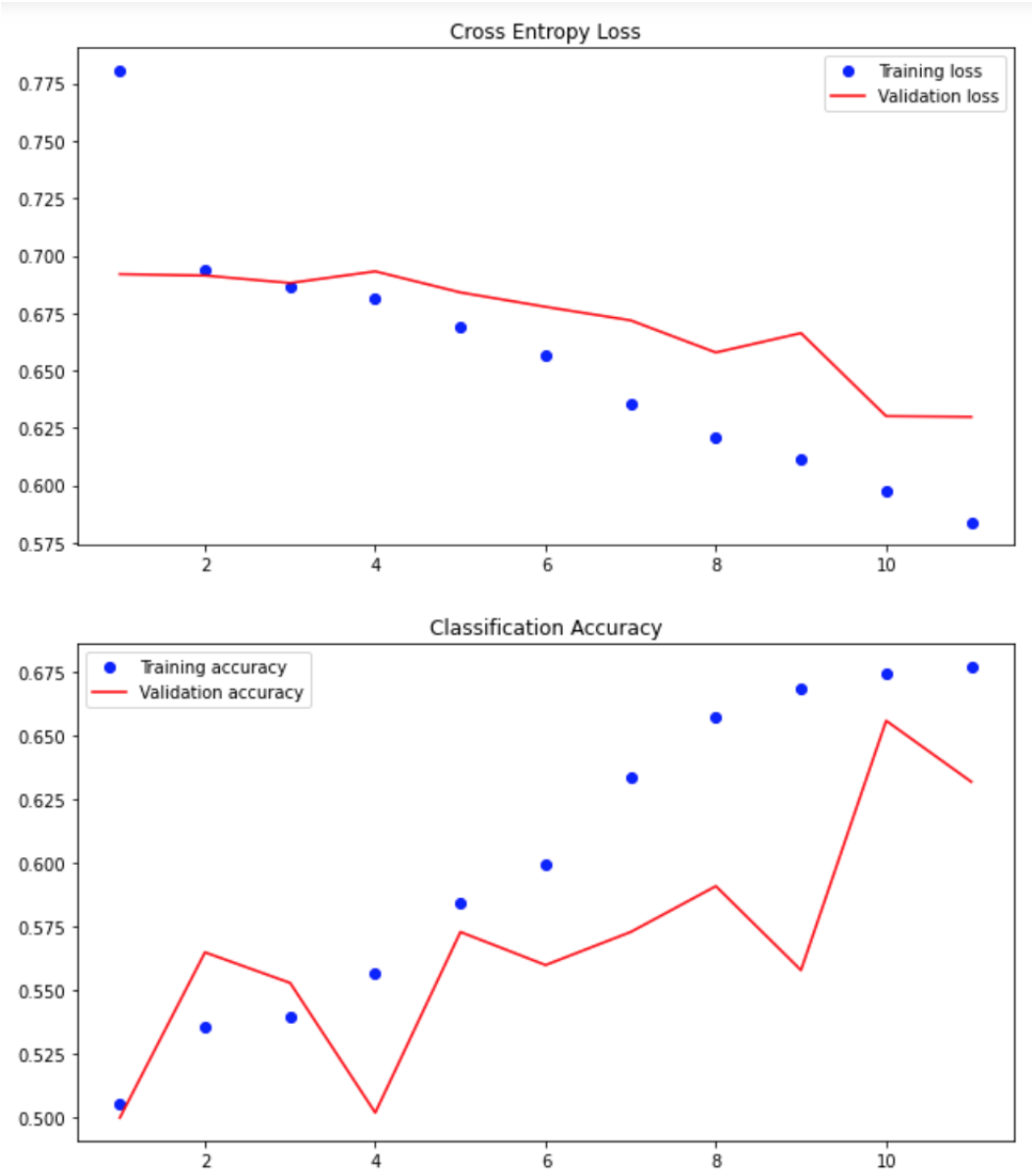
For an epoch of 30, the following learning curves are obtained.



(d) In this phase, the model was hypertuned by incorporating dropouts. The model summary and the learning curves are given below. Based on the new learning curves, the lowest stable validation loss occurred around the 10th epoch (from part c). However, all the learning curves, in general, showed similar behavior/trend. A possible reason may be the small number of training and validation data. The model is then saved as 'cats&dogs_1bcd.h5'.

Evaluation of test data gives:
>> Accuracy = 65.40 %

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 200, 200, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 100, 100, 32)      0
_________________________________________________________________
dropout (Dropout)            (None, 100, 100, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 98, 98, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 49, 49, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 47, 47, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 23, 23, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 23, 23, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 10, 10, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 12800)             0
_________________________________________________________________
dense (Dense)                (None, 512)               6554112
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 6,795,457
Trainable params: 6,795,457
Non-trainable params: 0
```



(g) The pre-trained convolution base of VGG16 has been used instead of our convolution layers. Note that the layers of the convolution base are frozen. The model summary of the VGG16 convolution base is shown below with the number of trainable weights before and after freezing.

VGG16 conv_base summary:

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 200, 200, 3)]	0
block1_conv1 (Conv2D)	(None, 200, 200, 64)	1792
block1_conv2 (Conv2D)	(None, 200, 200, 64)	36928
block1_pool (MaxPooling2D)	(None, 100, 100, 64)	0
block2_conv1 (Conv2D)	(None, 100, 100, 128)	73856
block2_conv2 (Conv2D)	(None, 100, 100, 128)	147584
block2_pool (MaxPooling2D)	(None, 50, 50, 128)	0
block3_conv1 (Conv2D)	(None, 50, 50, 256)	295168
block3_conv2 (Conv2D)	(None, 50, 50, 256)	590080
block3_conv3 (Conv2D)	(None, 50, 50, 256)	590080
block3_pool (MaxPooling2D)	(None, 25, 25, 256)	0
block4_conv1 (Conv2D)	(None, 25, 25, 512)	1180160
block4_conv2 (Conv2D)	(None, 25, 25, 512)	2359808
block4_conv3 (Conv2D)	(None, 25, 25, 512)	2359808
block4_pool (MaxPooling2D)	(None, 12, 12, 512)	0
block5_conv1 (Conv2D)	(None, 12, 12, 512)	2359808
block5_conv2 (Conv2D)	(None, 12, 12, 512)	2359808
block5_conv3 (Conv2D)	(None, 12, 12, 512)	2359808
block5_pool (MaxPooling2D)	(None, 6, 6, 512)	0

Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0

Total model summary after transferring the VGG16 conv_base:

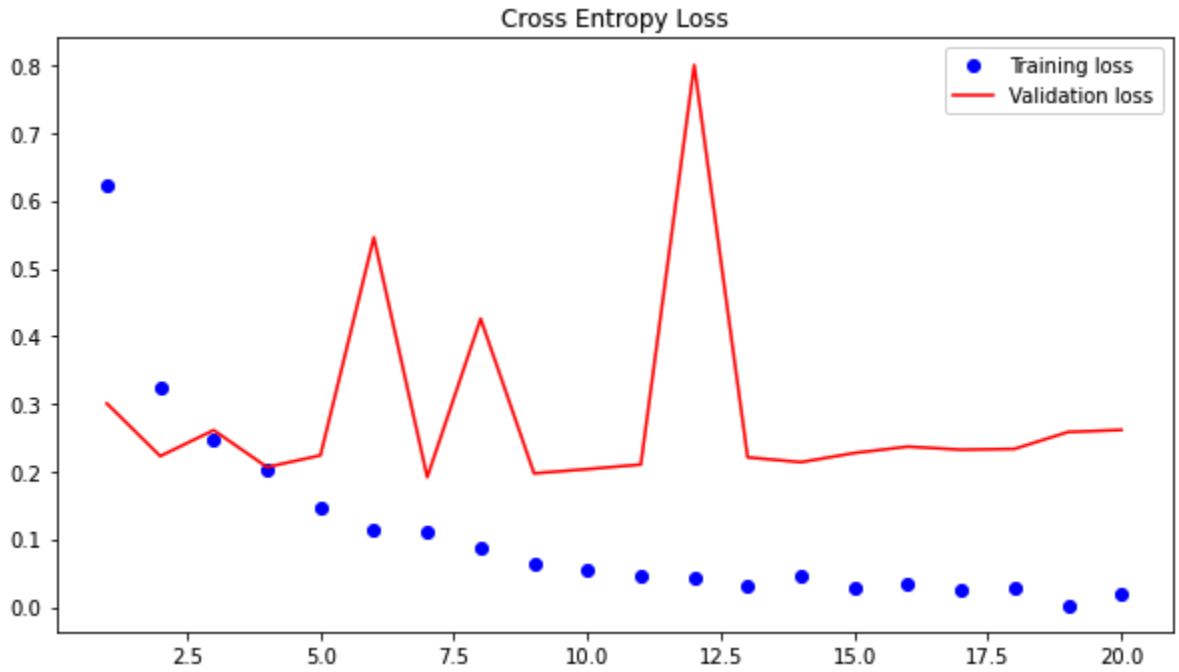
Model: "sequential"

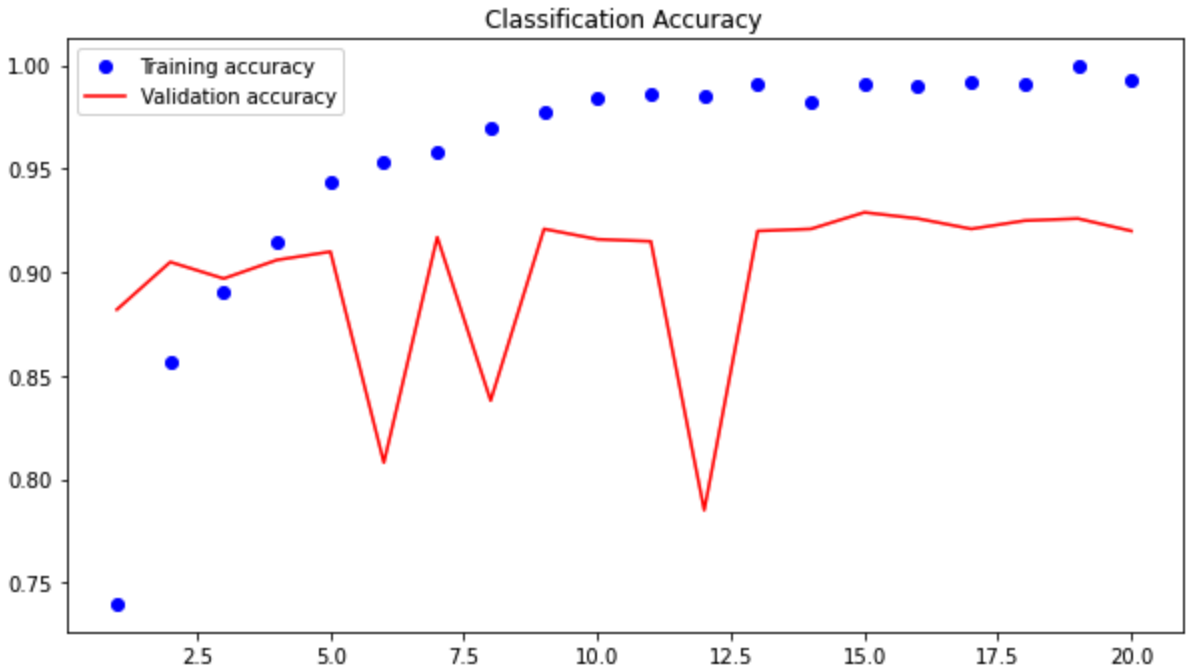
Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 6, 6, 512)	14714688
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 512)	9437696
dense_1 (Dense)	(None, 1)	513

Total params: 24,152,897
Trainable params: 24,152,897
Non-trainable params: 0

Number of weight tensors before freezing the conv_base = 30
Number of weight tensors after freezing the conv_base = 4

The learning curves are as follows:





It can be observed that the validation error decreased significantly, and the accuracy is now higher. Evaluation on test data (with 20 epoch) gives:
>> Accuracy = 92.55 %

The code is saved as ‘cats&dogs_1g.ipynb’ and the model is saved as ‘cats&dogs_VGG16_frozen.h5’.

(h) In this problem, the weights of the filters of all layers up to block 5 are freezed and the weights of the filters of the top most layer is kept unfreezed in order to fine tune the model.

Total model summary after transferring the VGG16 conv_base:

Model: "sequential"

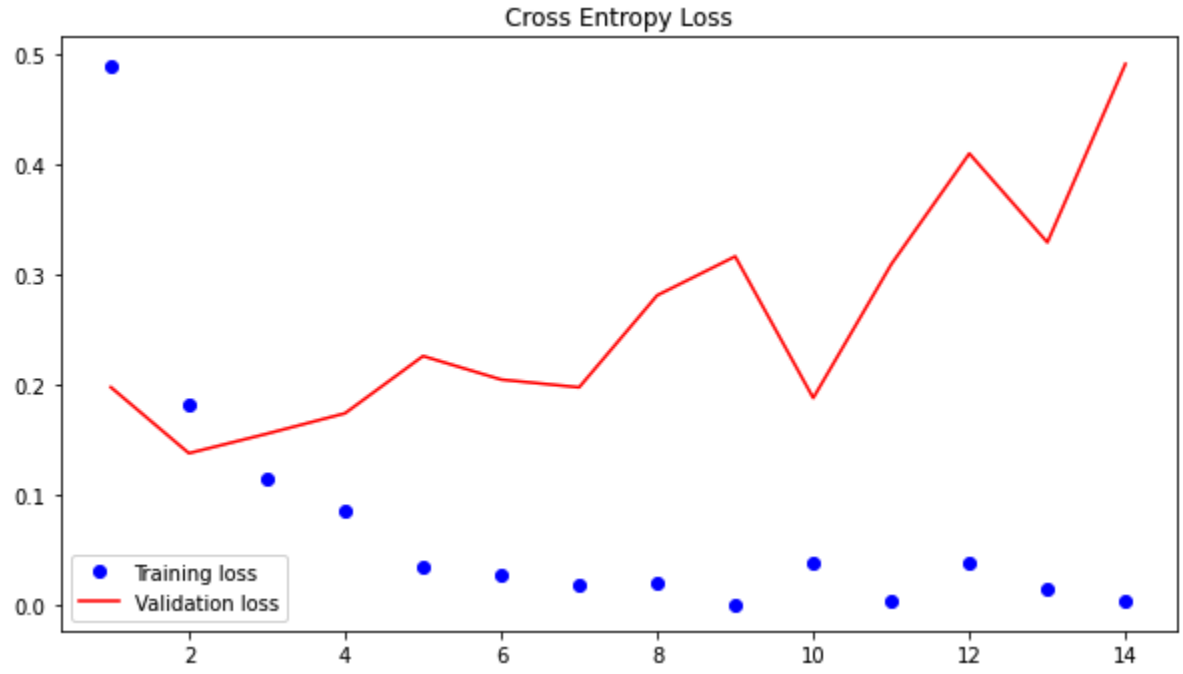
Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 6, 6, 512)	14714688
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 512)	9437696
dense_1 (Dense)	(None, 1)	513

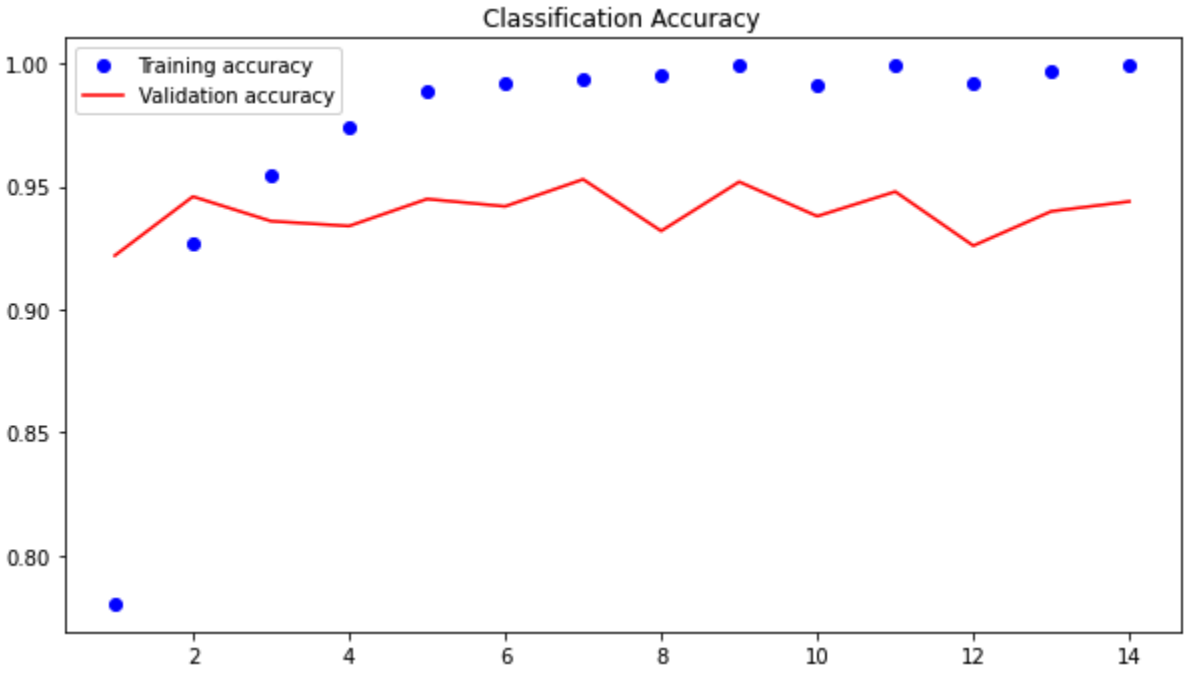
Total params: 24,152,897
Trainable params: 24,152,897

Non-trainable params: 0

Number of weight tensors before freezing the conv_base = 30
Number of weight tensors after freezing the conv_base = 10

The learning curves are as follows:





Both the training and validation loss decreased significantly as seen from the learning curves. At around 14 epoch, the lowest validation loss is observed. So, the number of epochs is fixed at 14, the model is saved as ‘cats&dogs_VGG16_unfrozen.h5’. The code is saved as ‘cats&dogs_1h.ipynb’. After evaluation of the saved model on the testing dataset, the accuracy found is:
>> Accuracy = 94.85 %

(i) This part continues from part 1(g), where the VGG16 convolution base was freezed. Note that in the previous cases, the number of training examples were limited but in this part, the training data generator is modified to perform data augmentation. This is performed to increase the number of training examples (through spatial shifting, rotation, and interpolation of the image pixels, etc.)

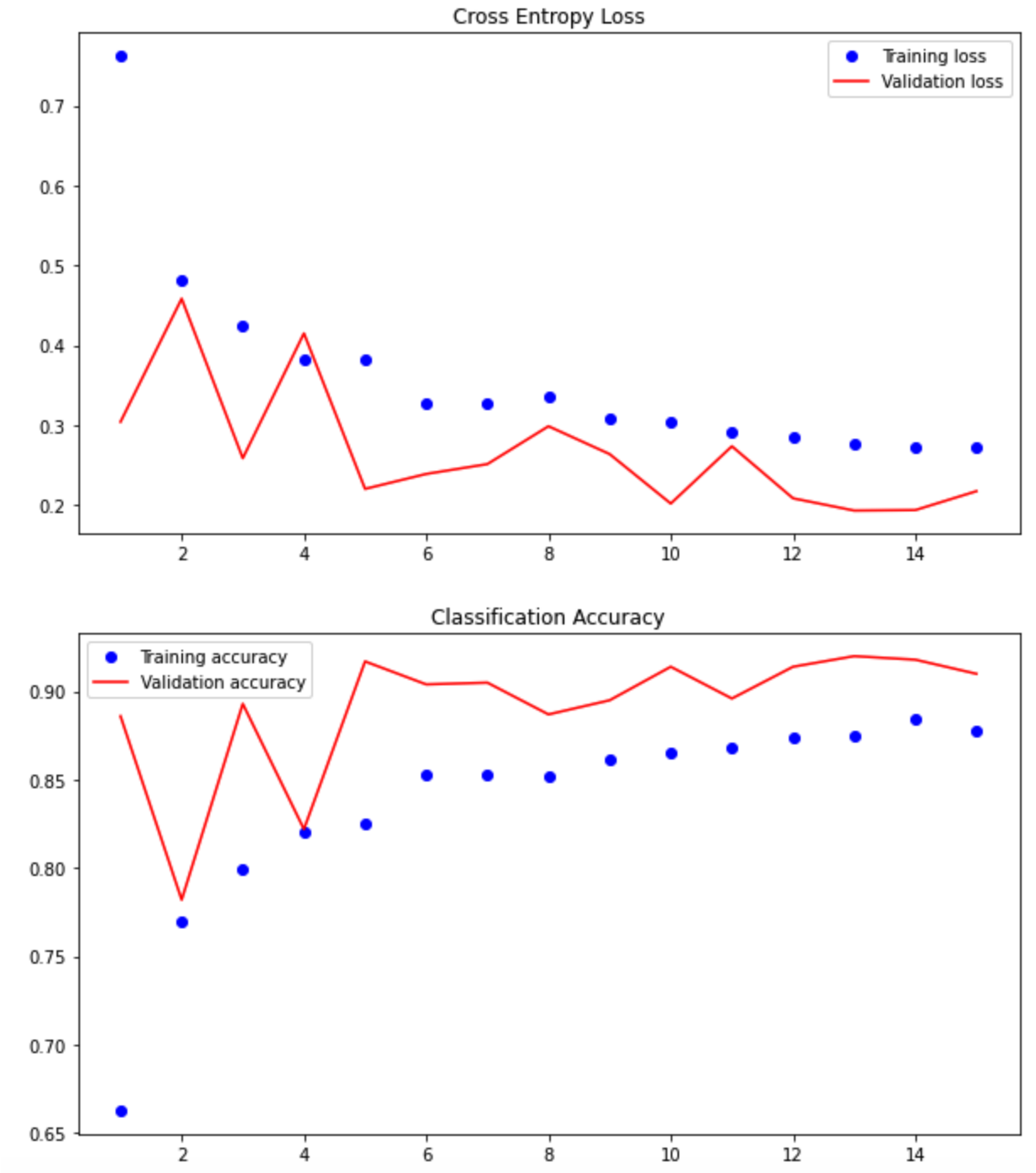
The model summary is as follows:

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 6, 6, 512)	14714688
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 512)	9437696
dense_1 (Dense)	(None, 1)	513

Total params: 24,152,897
Trainable params: 24,152,897
Non-trainable params: 0

The number of weight tensors before and after freezing the convolution base remains the same just as for part 1(g). The learning curves are shown below:



The validation loss reaches the minimum near the 14th epoch. It is clearly evident from the accuracy curve that the validation accuracy is much better than the one observed for part 1(g). So, keeping the number of epochs fixed at 14, the final model named ‘cats&dogs_VGG16_frozen_augment.h5’ is saved, and the code is saved as ‘cats&dogs_1i.ipynb’. Then the model was evaluated on the testing dataset, the accuracy found is:
>> Accuracy = 93.00 %

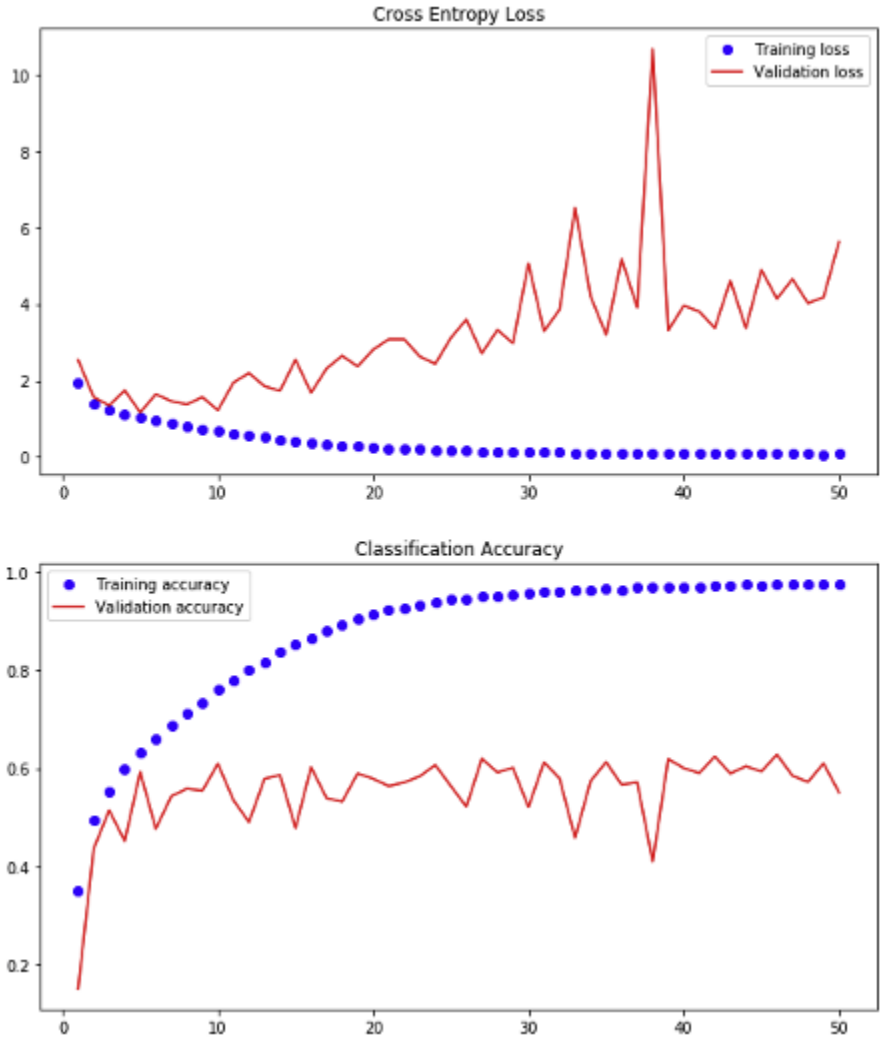
2. CIFAR-10

(a) Firstly, the Cifar-10 pickled dataset is downloaded from [2]. The dataset contains 60000 32x32 color images uniformly distributed to 10 classes, with 6000 images per class. The images are then unpickled from the data and test batches and reshaped to 32x32x3 size. Afterward, the images are normalized by dividing the image pixels by 255. The labels are also one-hot encoded.

(b) In this part, we have constructed a basic convolutional neural network with: 2D convolutional layers, max pooling layers, batch normalization, and one or more fully connected layers. As for activation functions, ReLu is used for the convolutional layers as well as fully connected layers. The 'Softmax' activation function is used only for the output layer. Moreover, the batch normalization is applied prior to the ReLu activation. The model summary is as follows:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               262272
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0
_________________________________________________________________
```

The model is compiled with RMSprop as the optimizer with a learning rate of 1e-4, categorical cross-entropy as the loss function to be optimized, and ‘accuracy’ as the metric. The learning curves for this model is given below:



After 8 epochs, the model overfits as seen from the curves.

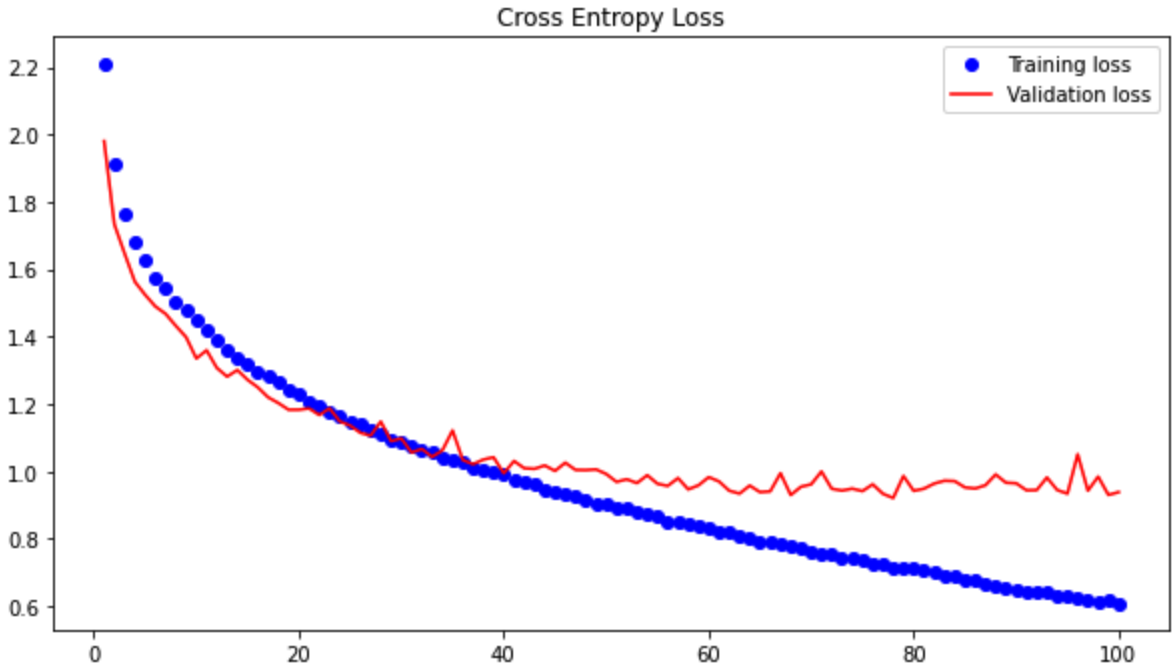
(c) In part 1(b), the validation accuracy was not very promising. So, here a closer examination of the network was done to fine tune the hyperparameters. The activation function is kept the same. The model summary is given below:

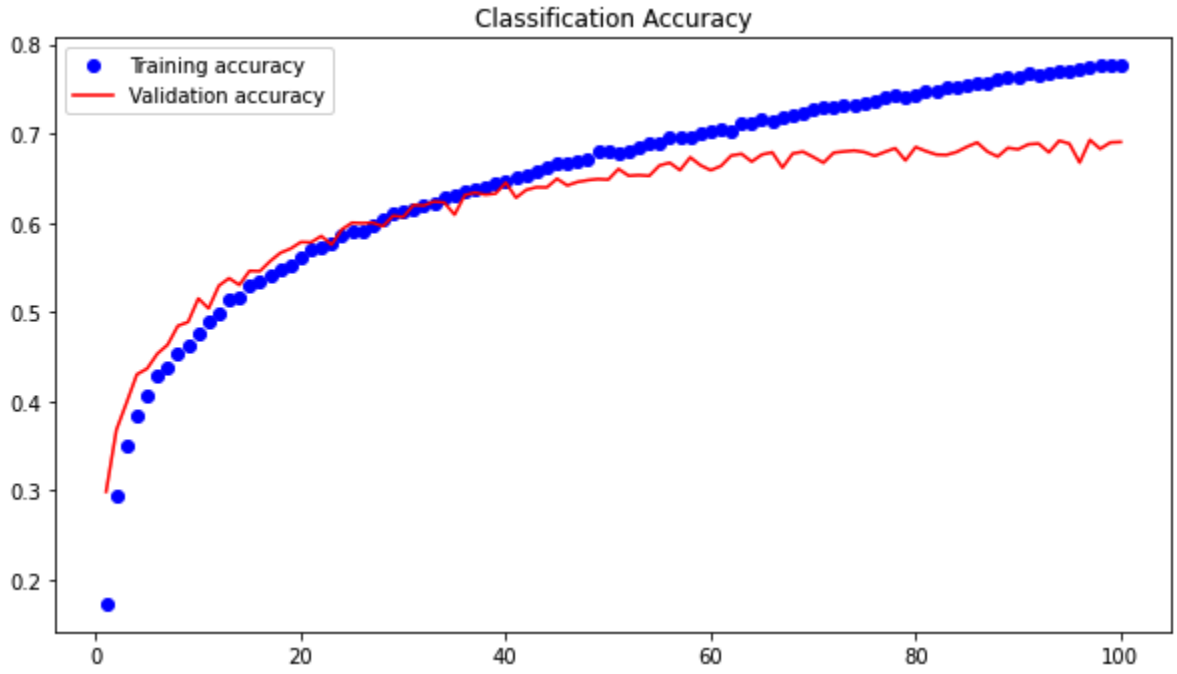
Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 64)	0
dropout_1 (Dropout)	(None, 8, 8, 64)	0
conv2d_4 (Conv2D)	(None, 8, 8, 128)	73856
conv2d_5 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 128)	0
dropout_2 (Dropout)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262272
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290

Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0

Here, we have used 3 VGG layers with a dropout of 0.2 after max pooling and also after the fully connected layers. This architecture resulted in the best performance. The model is compiled with stochastic gradient descent (SGD) as the optimizer, with a learning rate of 0.001 and a momentum of 0.9, in order to optimize the categorical cross-entropy loss. The network is run for 100 epochs. The learning curves are as follows:





The validation loss plateaus after 70 epochs as we can see from the curves above. The model accuracy is ~69% with 100 epochs on the testing dataset. The code is saved as 'cifar10_basic.ipynb', and the model is saved as 'cifar10_basic.h5'.

Evaluation on the test set (epoch = 100):
>> Accuracy = 69.07 %

(d) Two inception blocks were added after the two VGG blocks. Softmax activation is used for the output layer and all other layers use ReLu activation. Here, dropout is not incorporated. The model summary is as follows:

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 32, 32, 3)]	0	
conv2d (Conv2D)	(None, 32, 32, 32)	896	input_1[0][0]
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9248	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0	conv2d_3[0][0]
conv2d_7 (Conv2D)	(None, 8, 8, 64)	4160	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None, 8, 8, 64)	4160	max_pooling2d_1[0][0]
conv2d_8 (Conv2D)	(None, 8, 8, 64)	36928	conv2d_7[0][0]
conv2d_4 (Conv2D)	(None, 4, 4, 64)	4160	max_pooling2d_1[0][0]
conv2d_6 (Conv2D)	(None, 4, 4, 64)	36928	conv2d_5[0][0]

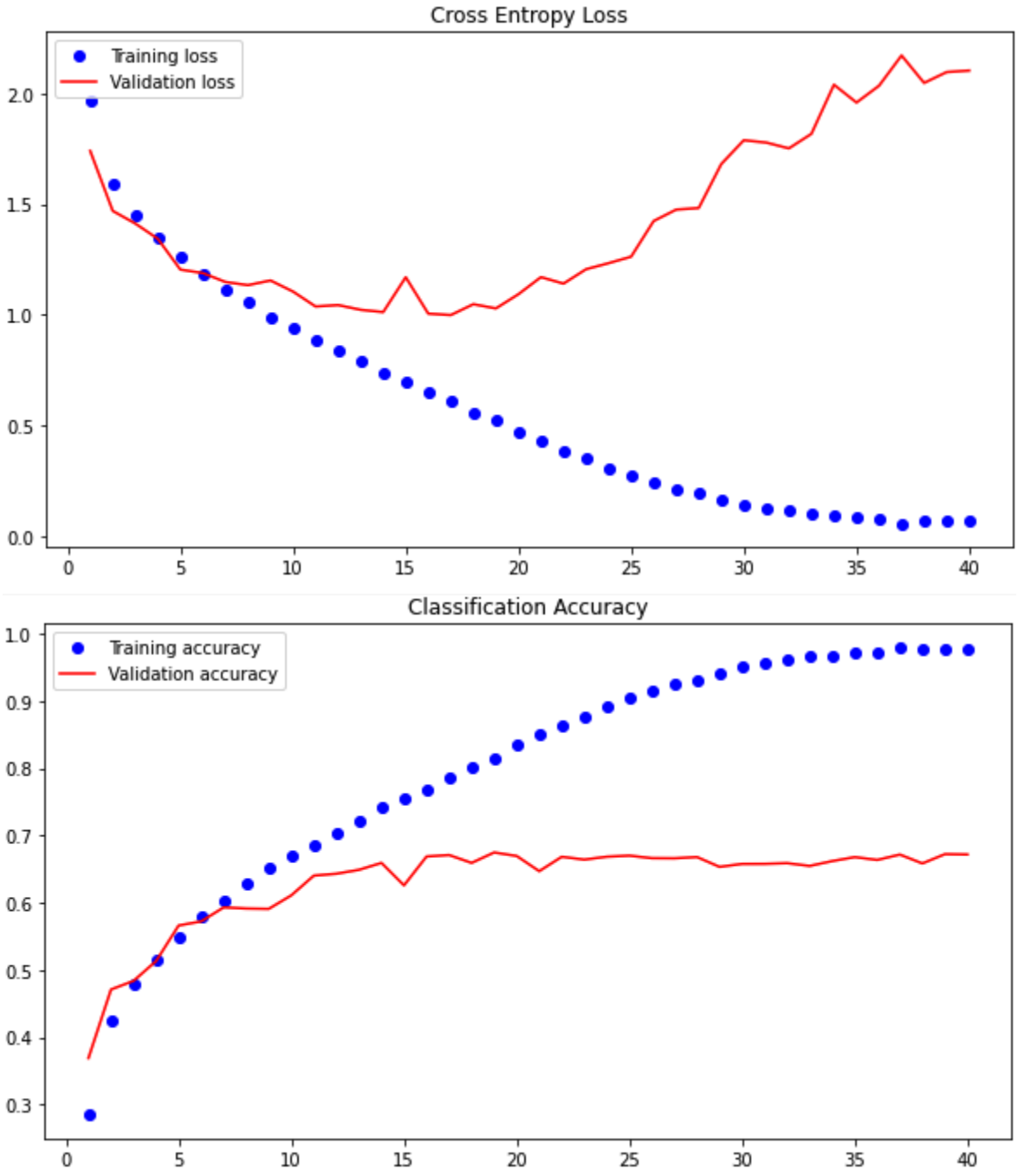
```
_____
conv2d_9 (Conv2D)               (None, 4, 4, 64)     36928       conv2d_8[0][0]
_____
_____
concatenate (Concatenate)       (None, 4, 4, 192)    0           conv2d_4[0][0]
                                                                 conv2d_6[0][0]
                                                                 conv2d_9[0][0]
_____
_____
conv2d_10 (Conv2D)              (None, 4, 4, 64)     12352       concatenate[0][0]
_____
_____
max_pooling2d_2 (MaxPooling2D)  (None, 4, 4, 192)    0           concatenate[0][0]
_____
_____
conv2d_11 (Conv2D)              (None, 4, 4, 64)     36928       conv2d_10[0][0]
_____
_____
conv2d_13 (Conv2D)              (None, 4, 4, 64)     102464      conv2d_9[0][0]
_____
_____
conv2d_14 (Conv2D)              (None, 4, 4, 64)     12352       max_pooling2d_2[0][0]
_____
_____
concatenate_1 (Concatenate)     (None, 4, 4, 192)    0           conv2d_11[0][0]
                                                                 conv2d_13[0][0]
                                                                 conv2d_14[0][0]
_____
_____
flatten (Flatten)               (None, 3072)         0           concatenate_1[0][0]
_____
_____
dense (Dense)                   (None, 128)          393344      flatten[0][0]
_____
_____
dense_1 (Dense)                 (None, 10)           1290        dense[0][0]
==================================================================================
==========
Total params: 747,562
Trainable params: 747,562
Non-trainable params: 0
```

This model has the same parameters like part 2(c). Observing the validation loss curve, the minimum loss is reached after 15 epochs whereas it took around 70 epochs for the convolutional network in part 2(c). This

shows that the gradient descent algorithm converged faster for the network which includes the two inception blocks than the one without the inception blocks. Then the testing dataset is evaluated by the model that was obtained after training the network for 15 epochs. The code is saved as ‘cifar10_inceptionblock.ipynb’.

Evaluation on the test set (epoch = 15):
>> Accuracy = 67.19 %

The learning curves are as follows:



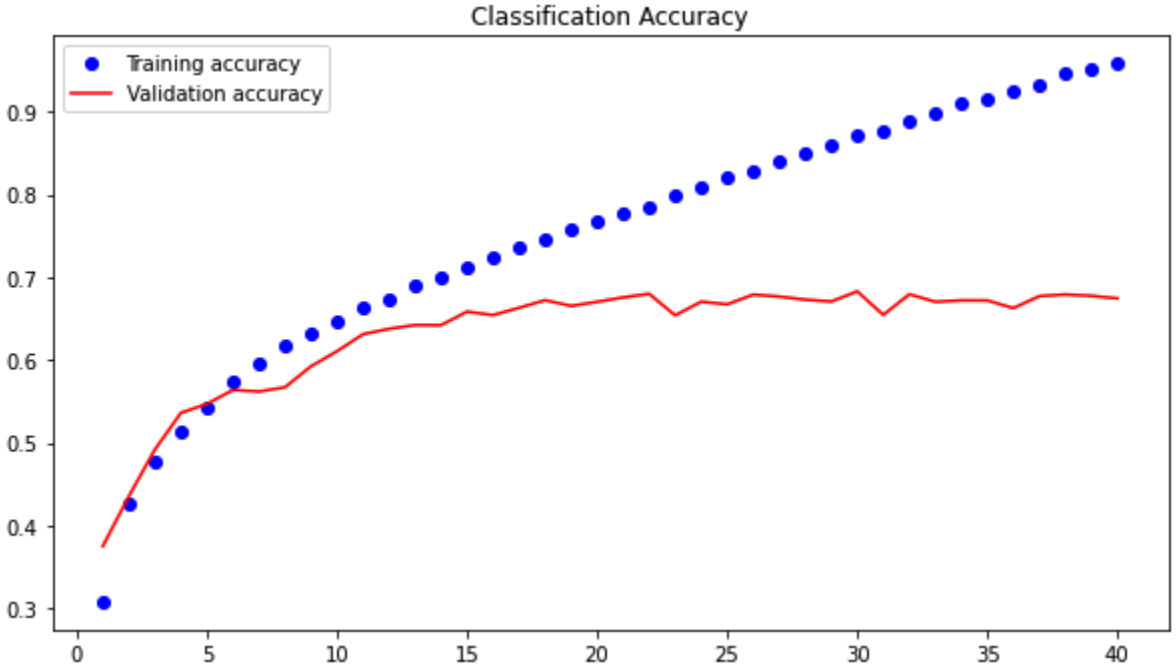
(e) In this part, 2 residual blocks are included after the 2 VGG blocks.Softmax activation is used for the output layer and ReLu activation function for all other layers. Here, dropout is not used. The model summary is given.

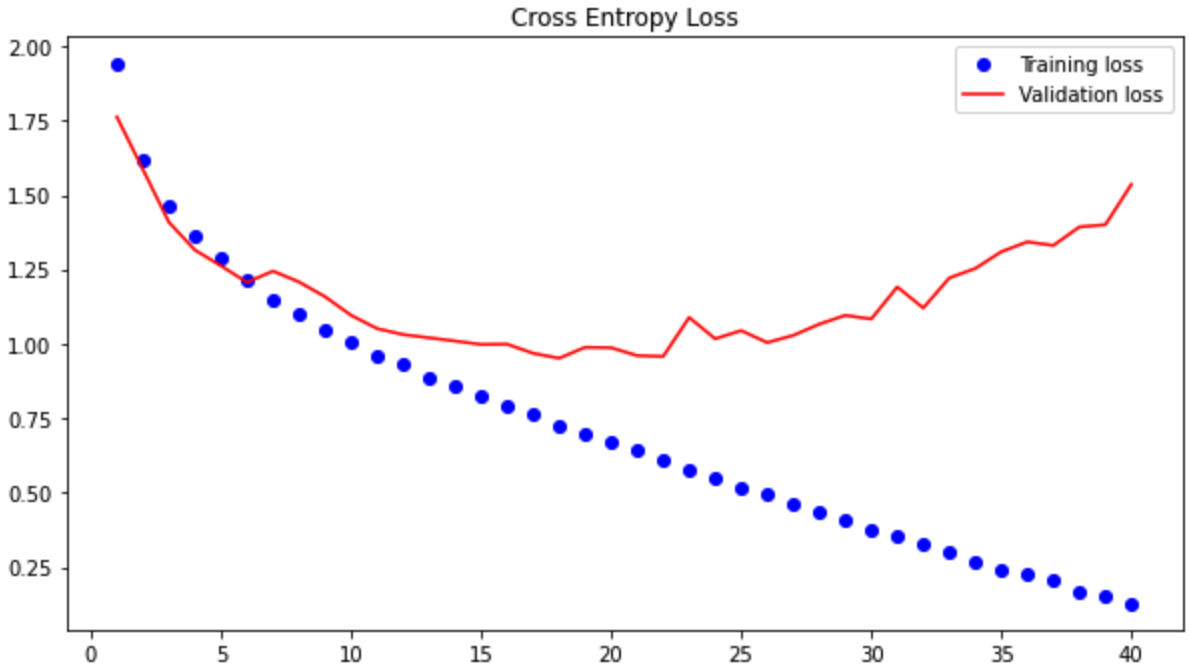
Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 32, 32, 3)]	0	
conv2d (Conv2D)	(None, 32, 32, 32)	896	input_1[0][0]
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9248	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 8, 8, 64)	36928	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None, 8, 8, 64)	36928	conv2d_4[0][0]
conv2d_6 (Conv2D)	(None, 8, 8, 64)	36928	conv2d_5[0][0]
add (Add)	(None, 8, 8, 64)	0	conv2d_6[0][0] max_pooling2d_1[0][0]
flatten (Flatten)	(None, 4096)	0	add[0][0]

```
_____
dense (Dense)                (None, 128)       524416      flatten[0][0]
_____
_____
dense_1 (Dense)              (None, 10)        1290        dense[0][0]
==================================================================================
==========
```

Total params: 702,058
Trainable params: 702,058
Non-trainable params: 0





The model is compiled with the same parameters like part 2(c). Based on the validation loss curve, the minimum loss is reached after 20 epochs whereas it took around 70 epochs for the convolutional network in part 2(c) and around 15 epochs for the model with inception block. Here, the gradient descent algorithm converged much later than the network with two inception blocks. Then, the test data is evaluated by selecting the model that was obtained after training the network for 20 epochs. The code is saved as 'cifar10_residualblock.ipynb'.

Evaluation on the test set (epoch = 20):
>> Accuracy = 67.48 %

Conclusion:
The best performance for the cifar10 classification is achieved with 2D convolutional blocks with residual block. Classification with inception block resulted in similar accuracy but slightly lower than that of residual block.

REFERENCES:
[1] Kaggle Cats and Dogs Dataset - https://www.microsoft.com/en-us/download/confirmation.aspx?id=54765
[2] Cifar-10 dataset - https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz