COL870

ASSIGNMENT 1

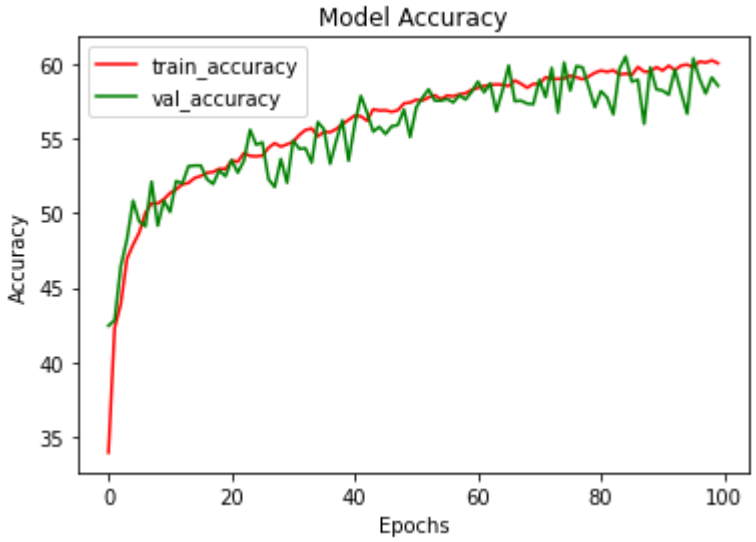
NIKKI TIWARI
2021SIY7560

PART (A) Effect of the following parameters on the classification performance

1)Number of convolutions layers

Model with 1 conv layer

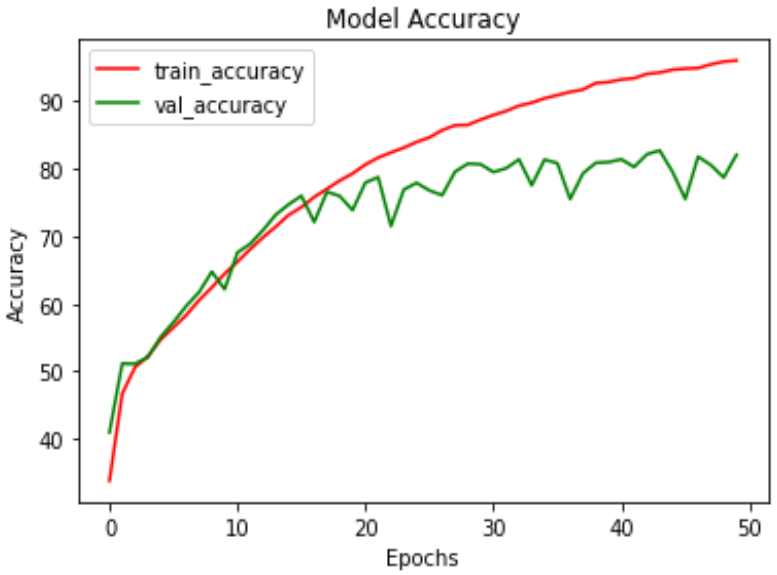
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 64, 64]           1,792
         MaxPool2d-2           [-1, 64, 32, 32]               0
            Linear-3                  [-1, 256]          16,640
            Linear-4                  [-1, 128]          32,896
            Linear-5                    [-1, 5]             645
================================================================
Total params: 51,973
Trainable params: 51,973
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 2.50
Params size (MB): 0.20
Estimated Total Size (MB): 2.75
----------------------------------------------------------------\
```



Accuracy vs Epochs plot for model with 1 conv layer

Model with 3 conv layers

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 64, 64]           1,792
         MaxPool2d-2          [-1, 64, 32, 32]               0
            Conv2d-3         [-1, 128, 32, 32]          73,856
         MaxPool2d-4         [-1, 128, 16, 16]               0
            Conv2d-5         [-1, 256, 16, 16]         295,168
         MaxPool2d-6           [-1, 256, 8, 8]               0
            Linear-7                 [-1, 256]          65,792
            Linear-8                 [-1, 128]          32,896
            Linear-9                   [-1, 5]             645
================================================================
Total params: 470,149
Trainable params: 470,149
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 4.38
Params size (MB): 1.79
Estimated Total Size (MB): 6.22
----------------------------------------------------------------
```

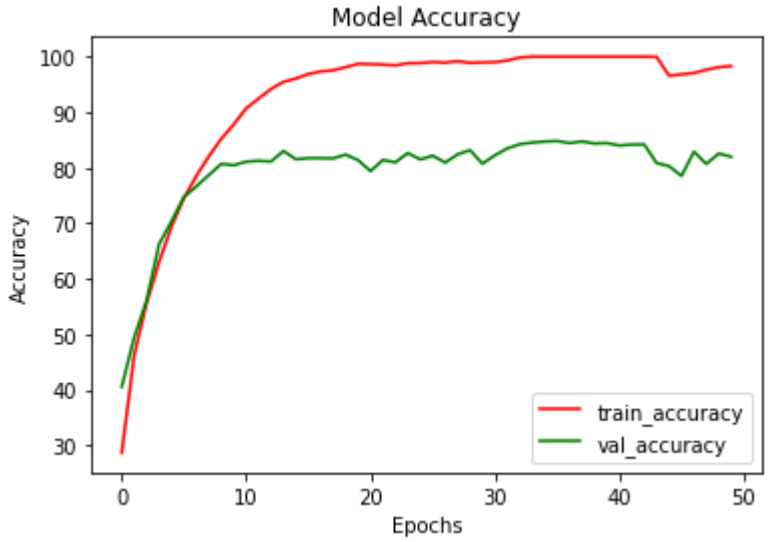


Accuracy vs Epchs plot for model with 3 conv layers

Model with 5 conv layers

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 64, 64]             896
         MaxPool2d-2          [-1, 32, 32, 32]               0
            Conv2d-3          [-1, 64, 32, 32]          18,496
         MaxPool2d-4          [-1, 64, 16, 16]               0
            Conv2d-5         [-1, 128, 16, 16]          73,856
         MaxPool2d-6           [-1, 128, 8, 8]               0
            Conv2d-7           [-1, 256, 8, 8]         295,168
         MaxPool2d-8           [-1, 256, 4, 4]               0
            Conv2d-9           [-1, 256, 4, 4]         590,080
        MaxPool2d-10           [-1, 256, 2, 2]               0
           Linear-11                 [-1, 256]          65,792
           Linear-12                 [-1, 128]          32,896
           Linear-13                   [-1, 5]             645
================================================================
Total params: 1,077,829
Trainable params: 1,077,829
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 2.39
Params size (MB): 4.11
Estimated Total Size (MB): 6.54
```

```
----------------------------------------------------------------
```

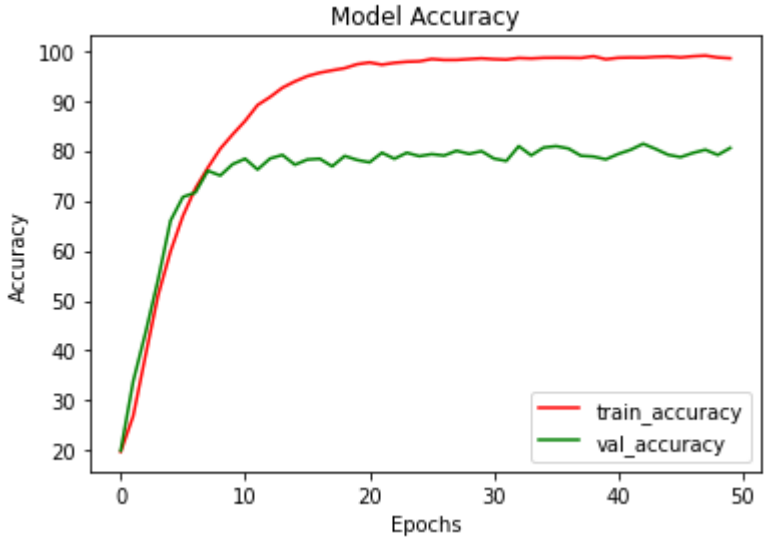


Accuracy vs Epchs plot for model with 5 conv layers

Model with 6 conv layers

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 64, 64]             896
         MaxPool2d-2          [-1, 32, 32, 32]               0
            Conv2d-3          [-1, 64, 32, 32]          18,496
         MaxPool2d-4          [-1, 64, 16, 16]               0
            Conv2d-5         [-1, 128, 16, 16]          73,856
         MaxPool2d-6           [-1, 128, 8, 8]               0
            Conv2d-7           [-1, 256, 8, 8]         295,168
         MaxPool2d-8           [-1, 256, 4, 4]               0
            Conv2d-9           [-1, 256, 4, 4]         590,080
        MaxPool2d-10           [-1, 256, 2, 2]               0
           Conv2d-11           [-1, 512, 2, 2]       1,180,160
        MaxPool2d-12           [-1, 512, 1, 1]               0
           Linear-13                 [-1, 256]         131,328
           Linear-14                 [-1, 128]          32,896
           Linear-15                   [-1, 5]             645
================================================================
Total params: 2,323,525
Trainable params: 2,323,525
Non-trainable params: 0
----------------------------------------------------------------
```

```
Input size (MB): 0.05
Forward/backward pass size (MB): 2.41
Params size (MB): 8.86
Estimated Total Size (MB): 11.32
----------------------------------------------------------------
```



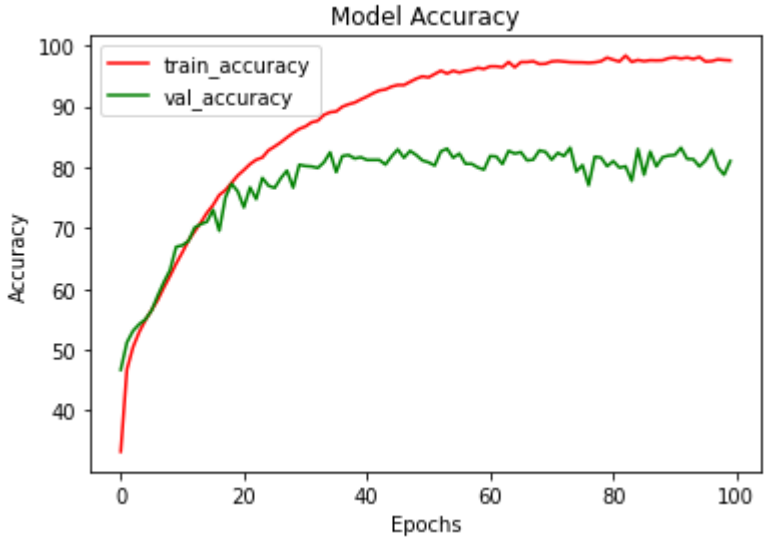
Accuracy(50 epochs)	train	val	test	parameters
conv1	60.076	58.57	59.28	51973
conv3	97.548	81	80.32	470149
conv5	98.312	81.96	80.25	1077829
conv6	98.44	80.52	78.96	2323525

Conclusion

In general as you increase the number of convolutional layers, we get better performance in train, val and test. However after a while we would reach a point wherein adding more conv layers may even slightly hurt the performance and result in very long training times as the number of parameters also increases.

Especially for the model with only 1 conv layer, we can understand that the model is under-fitting, and our model is not large enough to learn a good representation . However for the models with 5 and 6 conv layers, we can see that these models quickly overfit and get high training accuracy and low validation accuracy, this is because there are a large number of parameters .

2)**Number of epochs**



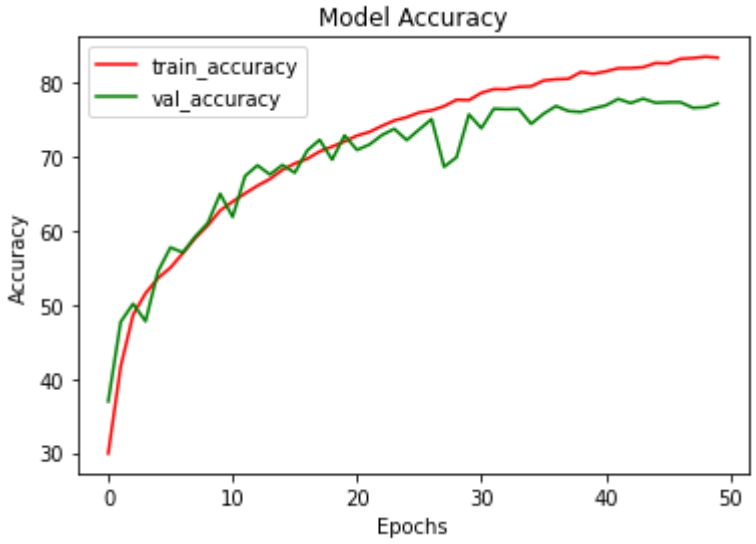
epochs	train	val	test
30	86.29	76.64	74.98
50	94.548	81.12	80.78
100	97.548	80.78	79.32

conclusion

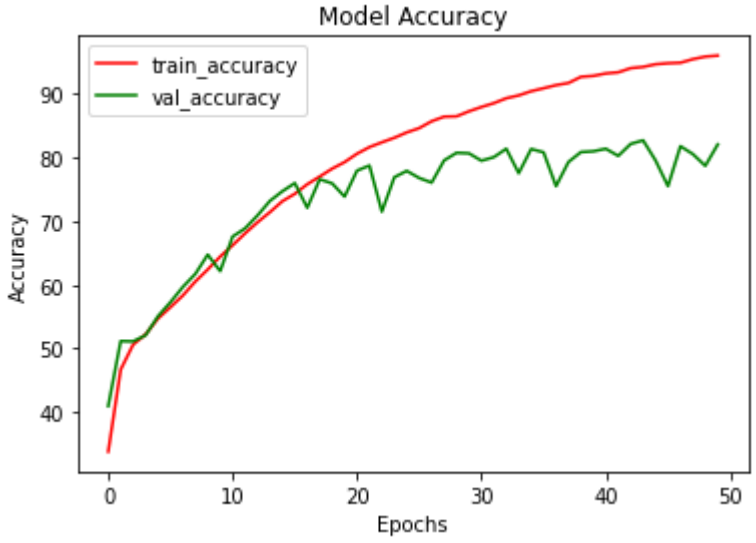
As the number of epochs increases , training set loss decreases and becomes nearly zero resulting in increase in training accuracy. Whereas, validation accuracy decreases depicting the overfitting of the model on training data.

3) Number of filters in different layers

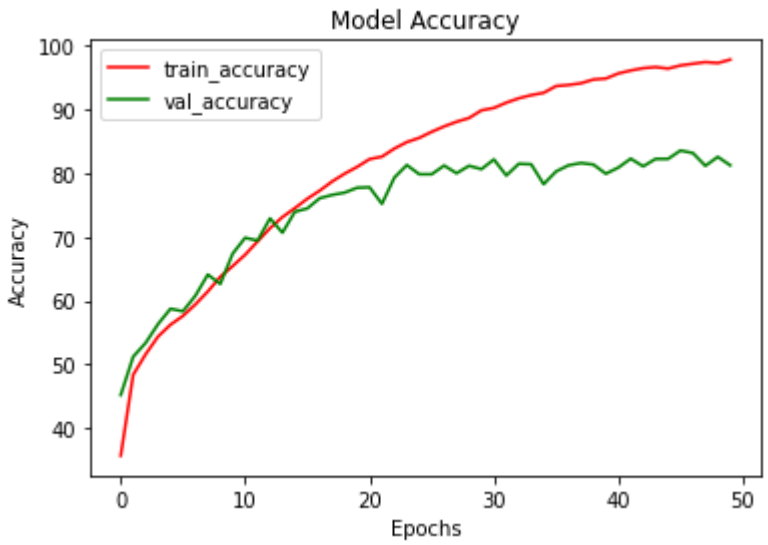
a)3 conv layers with 16,32,64 filters resp.



b) 3 conv layers with 64,256,256 filters p



c)3 conv layers with 256,512,1024 filters resp



conclusion

Each filter captures a pattern in the image . For example, the first layer of filters captures patterns like edges, corners, dots etc. Subsequent layers combine those patterns to make bigger patterns (like combining edges to make squares, circles, etc.).

Now as we move forward in the layers, the patterns get more complex; hence there are larger combinations of patterns to capture. That's why we increase the filter size in subsequent layers to capture as many combinations as possible.

But if we increase the number of filters after a certain point then our model would overfit as it would learn all the patterns and features in the train images and get low validation and test accuracies.

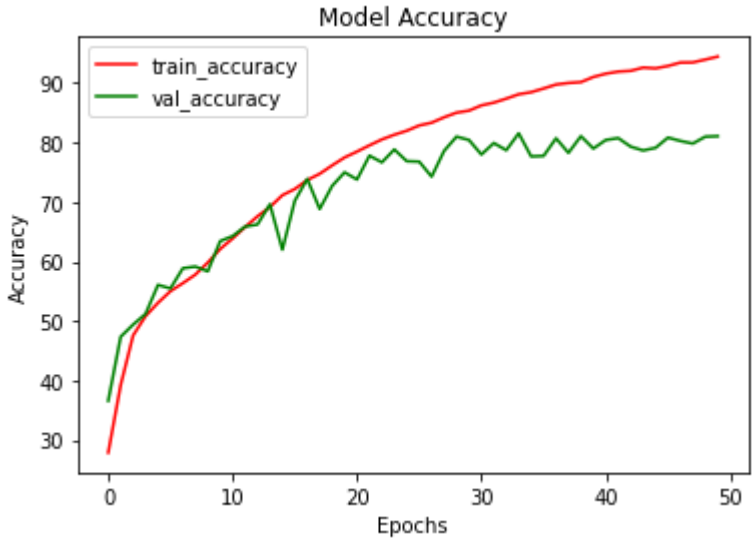
Model (50 epochs)	Train accuracy	Val accuracy	Test accuracy	Parameters
Filters-16,32, 64	83.332	77.160	76.520	73765

filters-64,256, 256	96.04	82.08	81.48	838917
Filters-256, 512,1024	97.76	81.20	80.40	6202885

4) fully connected (fc) layers

a)model with 4 FC layers

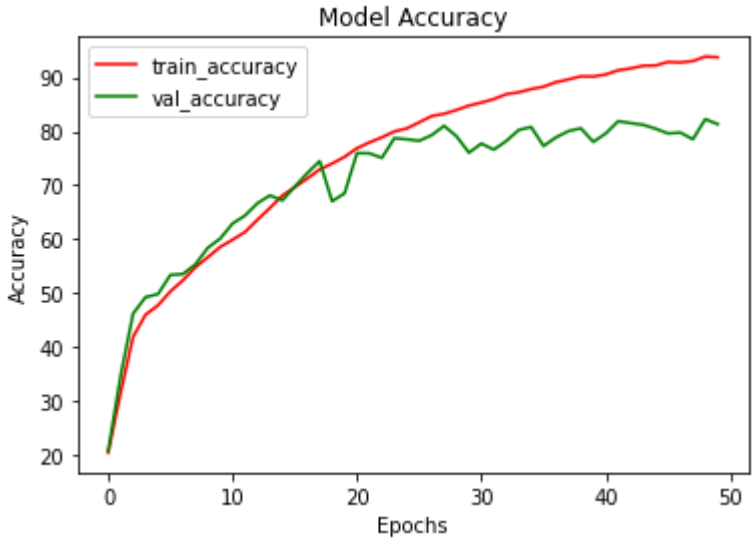
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 64, 64]           1,792
         MaxPool2d-2          [-1, 64, 32, 32]               0
            Conv2d-3         [-1, 128, 32, 32]          73,856
         MaxPool2d-4         [-1, 128, 16, 16]               0
            Conv2d-5         [-1, 256, 16, 16]         295,168
         MaxPool2d-6           [-1, 256, 8, 8]               0
            Linear-7                 [-1, 256]          65,792
            Linear-8                 [-1, 256]          65,792
            Linear-9                 [-1, 128]          32,896
           Linear-10                   [-1, 5]             645
================================================================
Total params: 535,941
Trainable params: 535,941
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 4.38
Params size (MB): 2.04
Estimated Total Size (MB): 6.47
----------------------------------------------------------------
```



Plot with 4 fully connected layers

b)model with 5 FC layers

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 64, 64]           1,792
         MaxPool2d-2          [-1, 64, 32, 32]               0
            Conv2d-3         [-1, 128, 32, 32]          73,856
         MaxPool2d-4         [-1, 128, 16, 16]               0
            Conv2d-5         [-1, 256, 16, 16]         295,168
         MaxPool2d-6           [-1, 256, 8, 8]               0
            Linear-7                 [-1, 512]         131,584
            Linear-8                 [-1, 256]         131,328
            Linear-9                 [-1, 256]          65,792
           Linear-10                 [-1, 128]          32,896
           Linear-11                   [-1, 5]             645
================================================================
Total params: 733,061
Trainable params: 733,061
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 4.38
Params size (MB): 2.80
Estimated Total Size (MB): 7.23
```

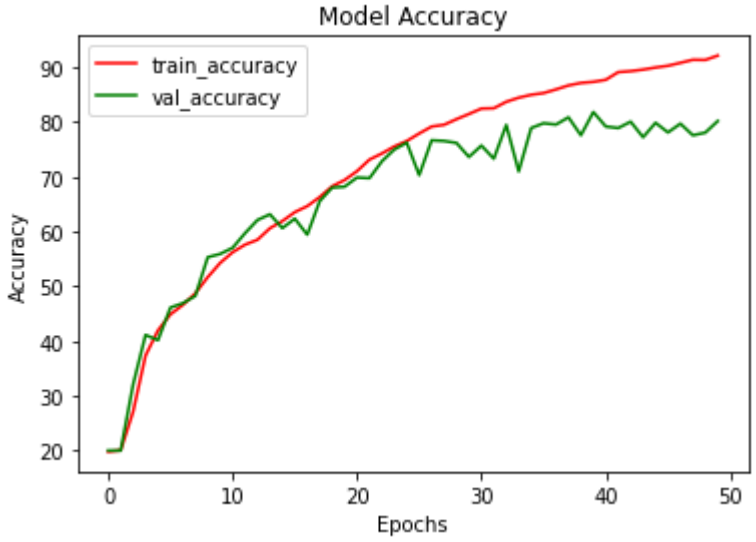


Plot with 5 Fully connected layers

c)Model with 6 FC layers

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 64, 64]           1,792
         MaxPool2d-2         [-1, 64, 32, 32]               0
            Conv2d-3        [-1, 128, 32, 32]          73,856
         MaxPool2d-4        [-1, 128, 16, 16]               0
            Conv2d-5        [-1, 256, 16, 16]         295,168
         MaxPool2d-6          [-1, 256, 8, 8]               0
            Linear-7                [-1, 256]          65,792
            Linear-8                [-1, 256]          65,792
            Linear-9                [-1, 256]          65,792
           Linear-10                [-1, 128]          32,896
           Linear-11                [-1, 128]          16,512
           Linear-12                  [-1, 5]             645
================================================================
Total params: 618,245
Trainable params: 618,245
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
```

Forward/backward pass size (MB): 4.38
Params size (MB): 2.36
Estimated Total Size (MB): 6.79

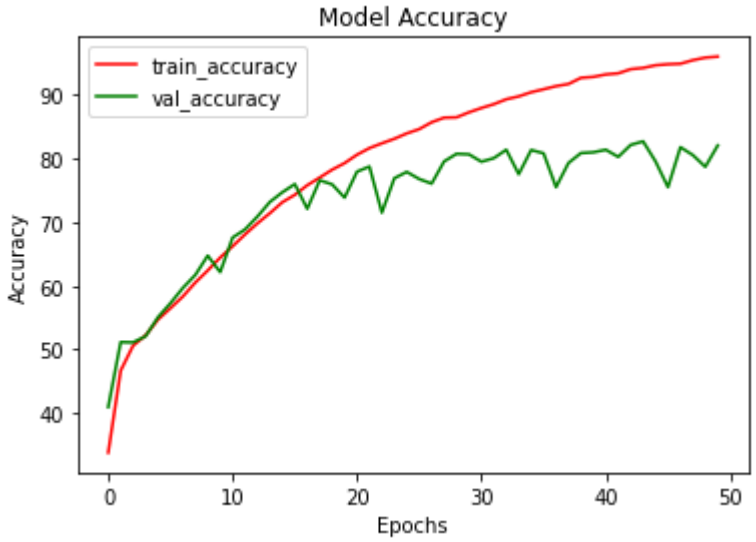


conclusion

Model	train	val	test
FC3	94.92	81.12	81.01
FC4	94.456	81.08	80.52
FC5	93.704	81.32	81.28
FC6	92.15	80.2	78.8

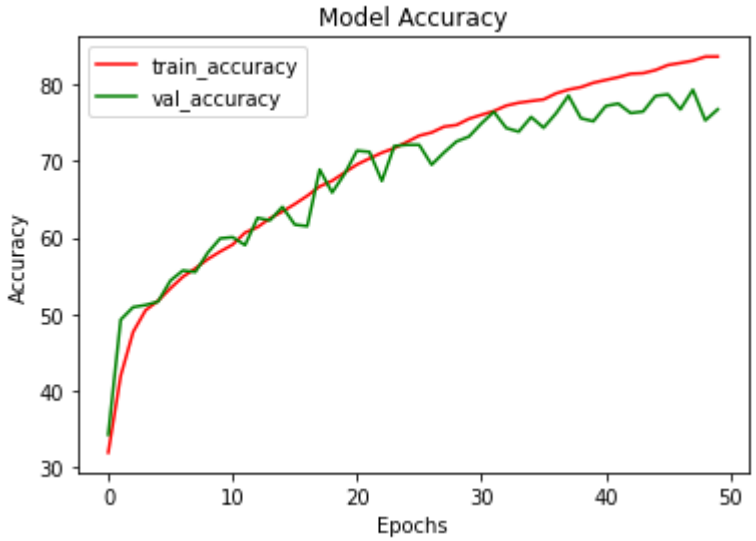
5)Max pooling layers

a)model with consecutive max pooling layers



Plot for Model with max pooling layer after every conv layer

b)model with alternate max pooling layers



Plot for alternated max pooling layers

Consecutive max pooling layers	94.92	81.12	80.78

Alternate Max pooling in layers	83.664	76.76	74.96

Conclusion

In CNN the output feature maps are sensitive to the location of features in the input. If the input image is translated the output feature map will also be affected by the translation, so that small movements in the position of the feature in the input image will result in a different feature map. One way to address this sensitivity problem is using pooling layers, because of their down sampling ability. Pooling layers create a lower resolution version of the input that still contains the large or important structural elements, without the fine details which may be not useful for the task.

So the max pooling layer makes the image unclear for the human eye by sampling it down to a lower resolution, but for the machine learning model it mostly removes not relevant elements and makes it more robust to changes in the input (like rotation, shifting, translation etc.)

They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model's invariance to local translation.
Pooling mainly helps in extracting sharp and smooth features. It is also done to reduce variance and computations. Max-pooling helps in extracting low-level features like edges, points, etc. While Avg-pooling goes for smooth features.

As we can see in the results that in alternate max pooling layers if features of images were slightly changed in their location the model was not able to classify the object .
Moreover max pooling layers also does not increase the number of parameters.
Thus number of parameters is the same in both the models .

6)Stride

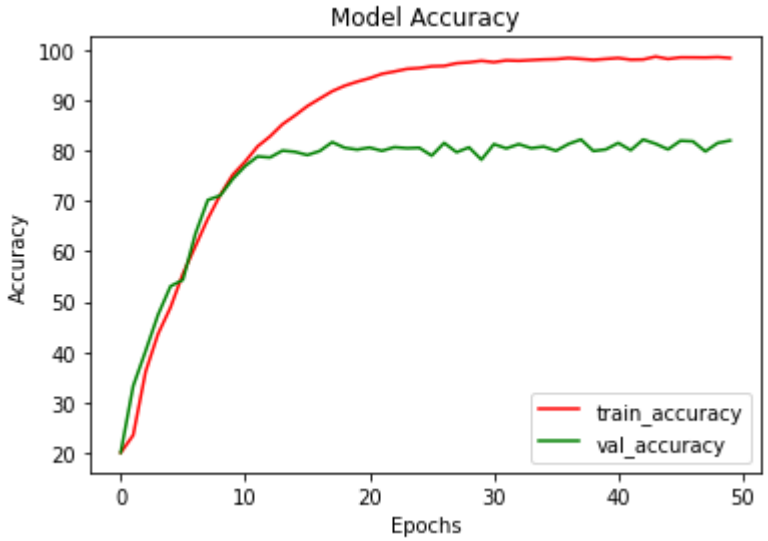
Convolving with stride ≥ `1` seems to result in a reduced accuracy PROBABLY because we miss out on features when increasing stride.
Increasing stride leads to loss of information. If a pattern is visible most strongly on a position that is skipped, that information is lossed. To avoid this, usually max pooling layers are used to shrink the size of activation maps. Of course, they still lose some information about the exact location of an observed feature.

BEST MODEL

Accuracy of the network on the train images: 98.28000 %

Accuracy of the network on the test images: 83.520000 %
performing test

Accuracy of the network on the test images: 81.720000 %
Accuracy of aeroplane :88.000000 %
Accuracy of bird : 73.000000 %
Accuracy of cat : 84.00000 %
Accuracy of dog : 73.800000 %
Accuracy of ship : 94.400000 %




```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 64, 64]           1,792
```

```
         MaxPool2d-2           [-1, 64, 32, 32]               0
            Conv2d-3          [-1, 128, 32, 32]          73,856
         MaxPool2d-4          [-1, 128, 16, 16]               0
            Conv2d-5          [-1, 256, 16, 16]         295,168
         MaxPool2d-6            [-1, 256, 8, 8]               0
            Conv2d-7            [-1, 512, 8, 8]       1,180,160
         MaxPool2d-8            [-1, 512, 4, 4]               0
            Linear-9                  [-1, 128]          65,664
           Dropout-10                  [-1, 128]               0
           Linear-11                  [-1, 256]          33,024
           Dropout-12                  [-1, 256]               0
           Linear-13                  [-1, 512]         131,584
           Dropout-14                  [-1, 512]               0
           Linear-15                    [-1, 5]           2,565
================================================================
Total params: 1,783,813
Trainable params: 1,783,813
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 4.70
Params size (MB): 6.80
Estimated Total Size (MB): 11.55
----------------------------------------------------------------
```

IMages Mis-classified by the model

Out of 2500 test images ,the model mis-classifies 463 images
This is the per-class accuracies
Accuracy of aeroplane :88.000000 %
Accuracy of bird : 73.000000 %
Accuracy of cat : 84.00000 %
Accuracy of dog : 73.800000 %
Accuracy of ship : 94.400000 %

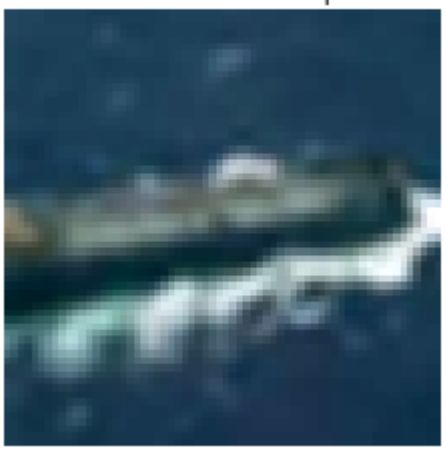
Reasons for misclassification

1)Most of the time model misclassified due to **imperfect dataset**

- **Intra-Class Variation.**
- **Scale Variation.**
- **View-Point Variation.**
- **Occlusion.**
- **Illumination.**
- **Background Clutter.**

2)Model's inability to distinguish between **common/similar inter-class features** for ex -model misclassifies ship as aeroplane and aeroplane as ship as both of them have similar shape and structure .The model also misclassifies the dog as cat ,whenever image is not in its front view or the face is not very clear, as both of them has many common features and similar shape and size.

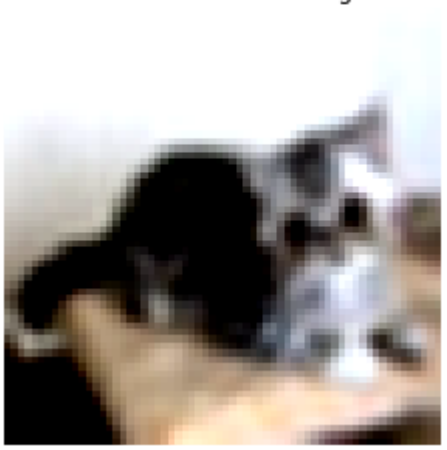
True Class--ship
Predicted Class-- aeroplane



True Class--cat
Predicted Class-- dog

True Class--bird
Predicted Class-- aeroplane

True Class--cat
Predicted Class-- dog



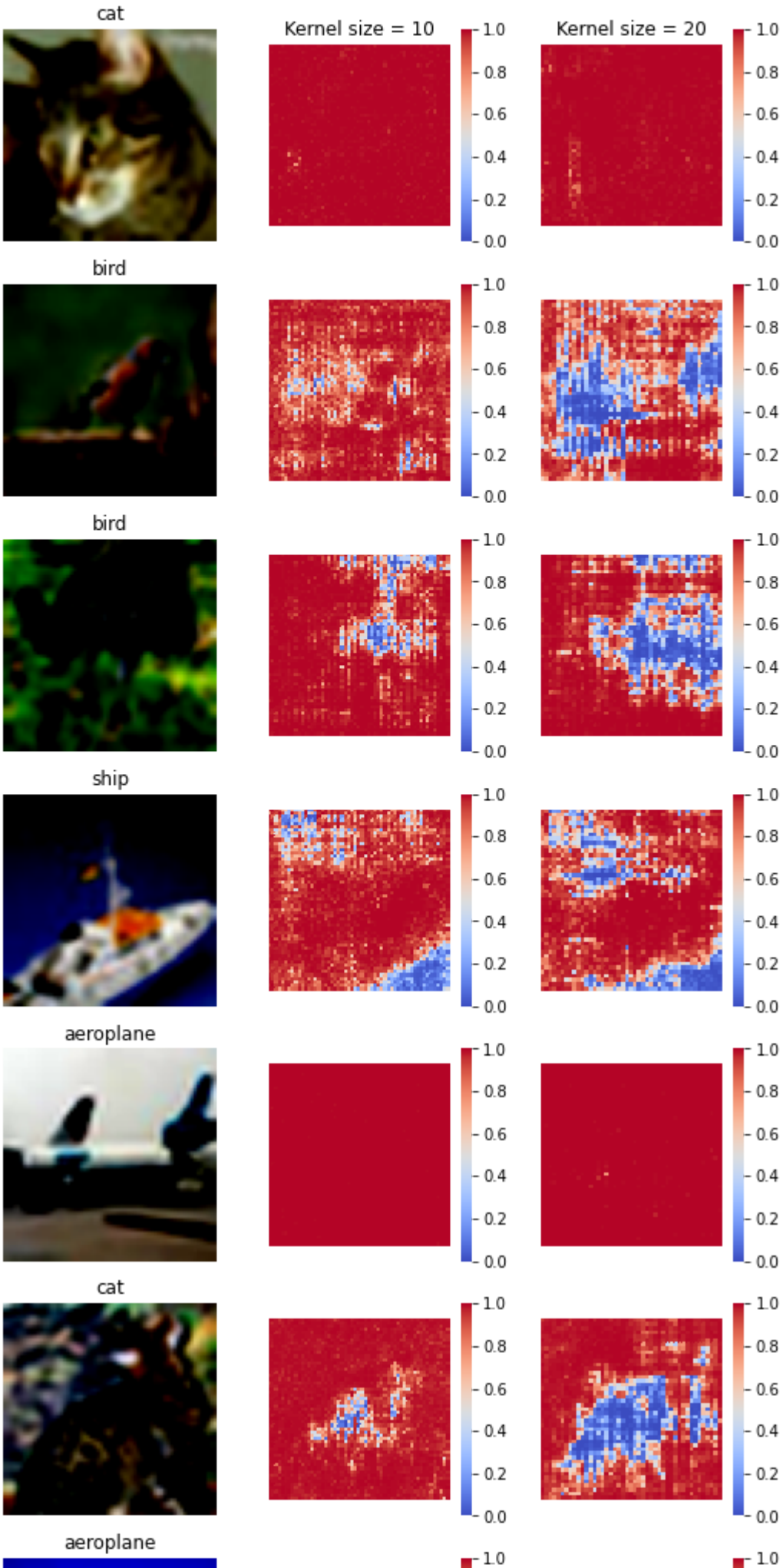
True Class--aeroplane
Predicted Class-- ship

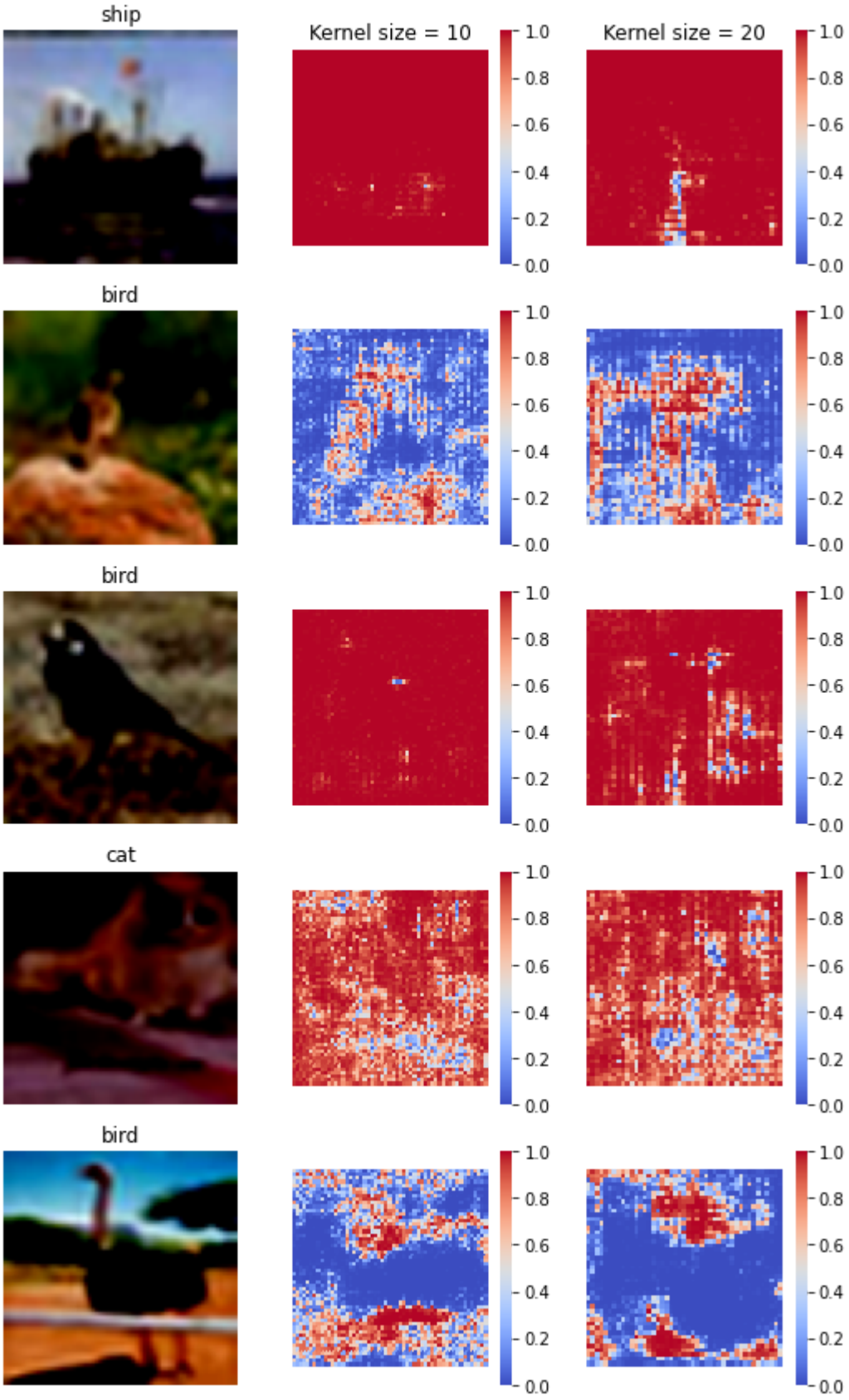
True Class--dog
Predicted Class-- aeroplane

Part (B)

2.1 Occlusion sensitivity experiment

On a fully-trained high-performance image classifier model(PART-A), to answer whether the model has really learnt the location of the object in the image or if the model just classifies the image based on surrounding or contextual cues.
We had to perform occlusion sensitivity experiment on 10 images from the test set by following these steps (as mentioned in the assignment)
For each pixel position i(along x-direction), j (along y-direction):
1) consider a window (N × N , choose an appropriate value of N) around (i, j) and replace the content of the window with grey pixels.
2)Pass the modified image (with respect to the position (i, j)) through the model and note down the
probability for true class into an array. i.e., confidence(i, j).
3) Plot the confidence array as an image an


cat
Kernel size = 10
Kernel size = 20
1.0
0.8
0.6
0.4
0.2
0.0
bird
bird
ship
aeroplane
cat
aeroplane



Images with probability heatmaps for kernels of size 10 × 10 and 20 × 20 for the occlusion sensitivity experiment

Conclusions from Occlusion sensitivity experiment

1)We can conclude that the model is truly identifying the location of the object in the image, and not just using the surrounding context. We attempted to conclude this by systematically occluding different portions of the input image with a grey square, and monitoring the output of the classifier. The examples clearly show that **the model is localising the objects within the scene**, as **the probability of the correct class drops significantly when the object is occluded**.

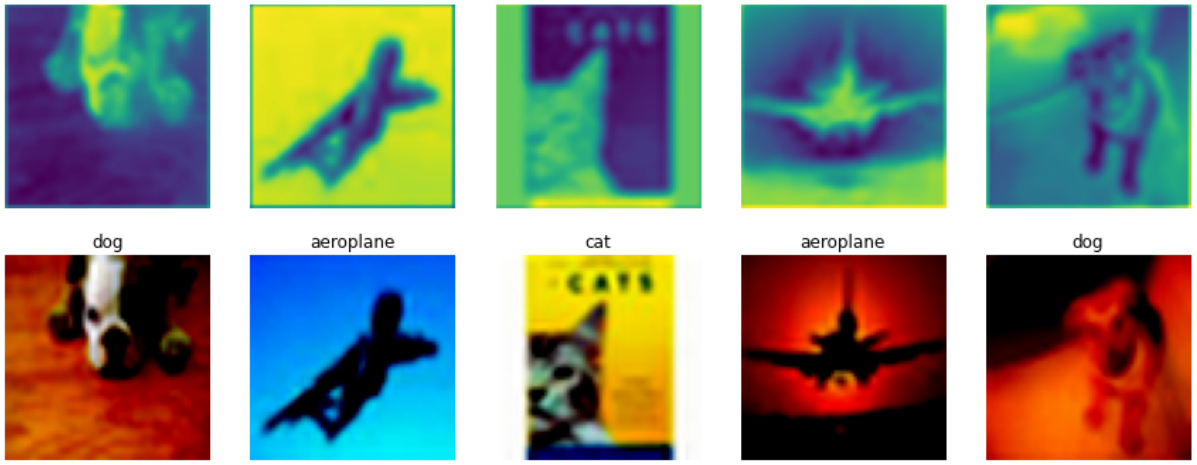
2)When the occluder covers the image region that appears in the visualisation, we see a strong drop in activity in the feature map. This shows that the visualisation genuinely corresponds to the image structure that stimulates that feature map.

3)Occlusion experiments were carried on with window sizes 10×10 and 20×20 respectively.
From the heatmap outputs, **we can infer that increasing the occlusion kernel size around the main features of the objects to be classified causes the probability of misclassification to increase (increase in blue shading) when the red areas are occluded**, **the score for class goes down.**

2.2 Filter Analysis

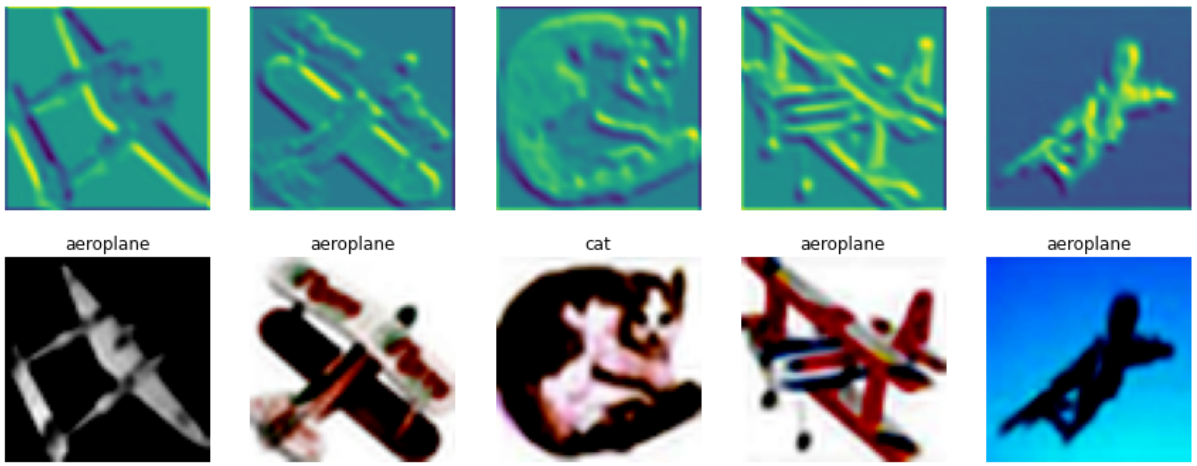
2.2.1 Filter Identification

conv1_filter3



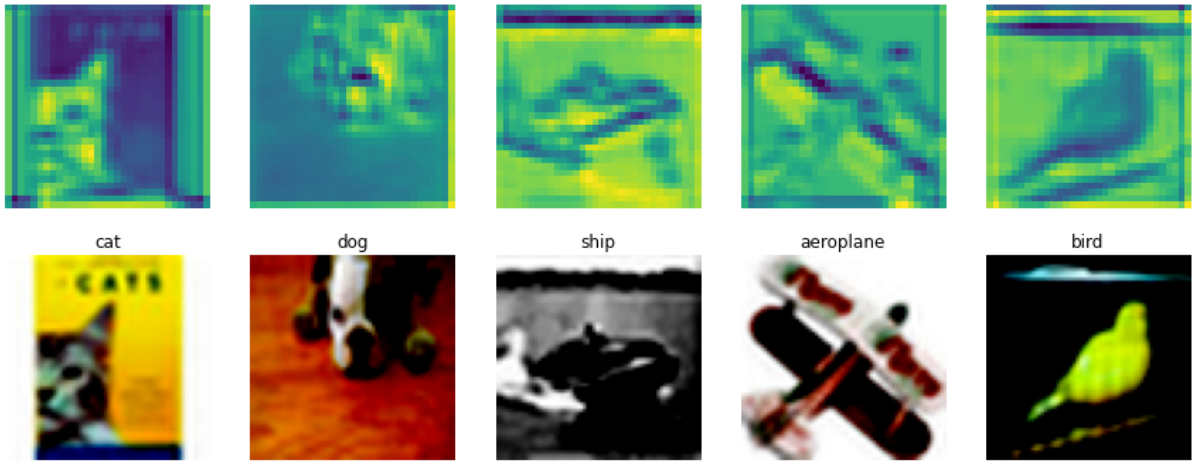
Output of 3rd filter of 1st conv layer

conv1_filter31

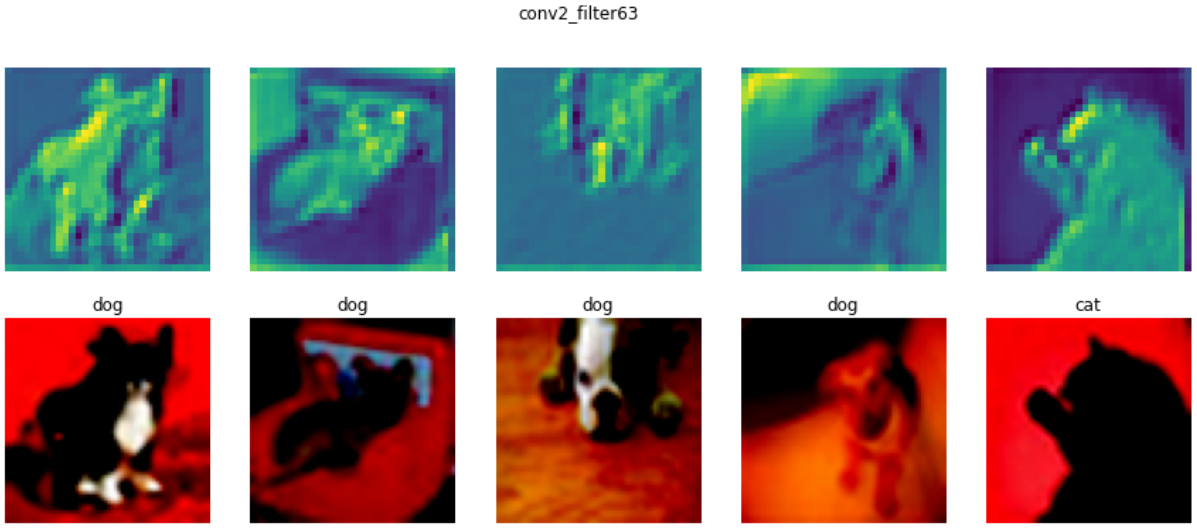


Output of 31st filter of 1st conv layer

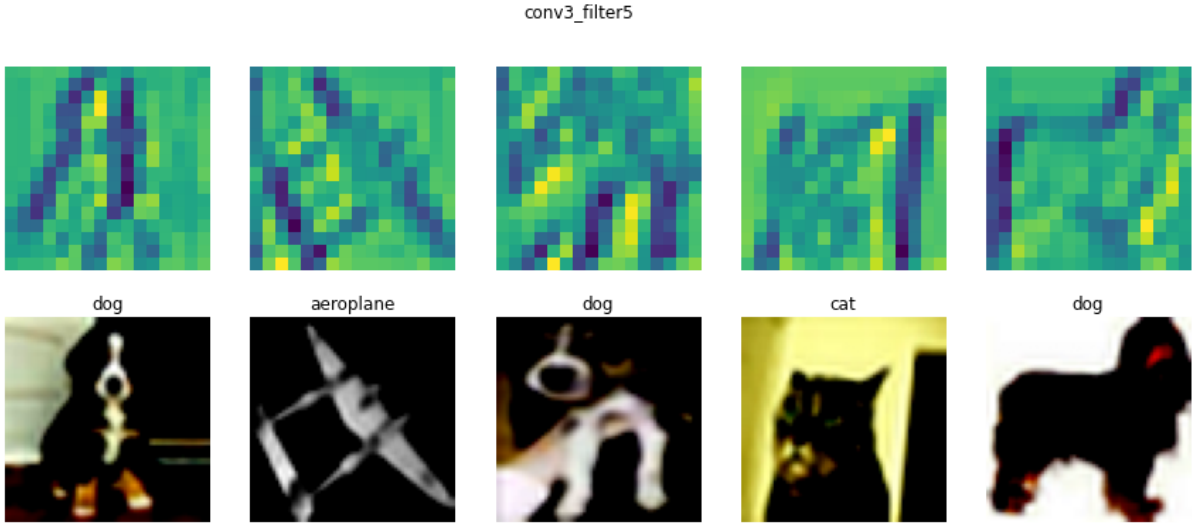
conv2_filter4



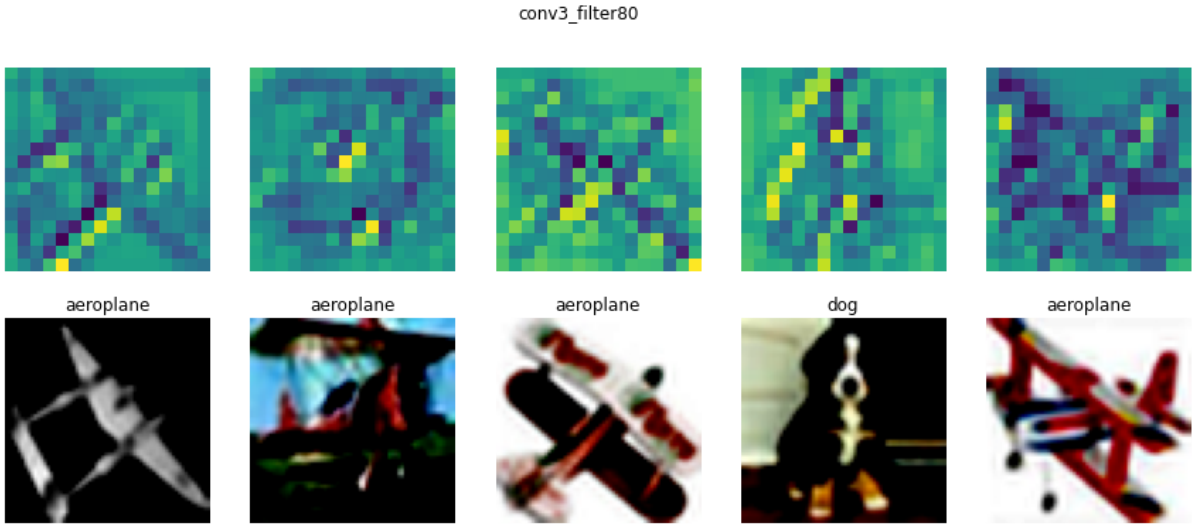
Output of 4th filter of 2nd conv layer



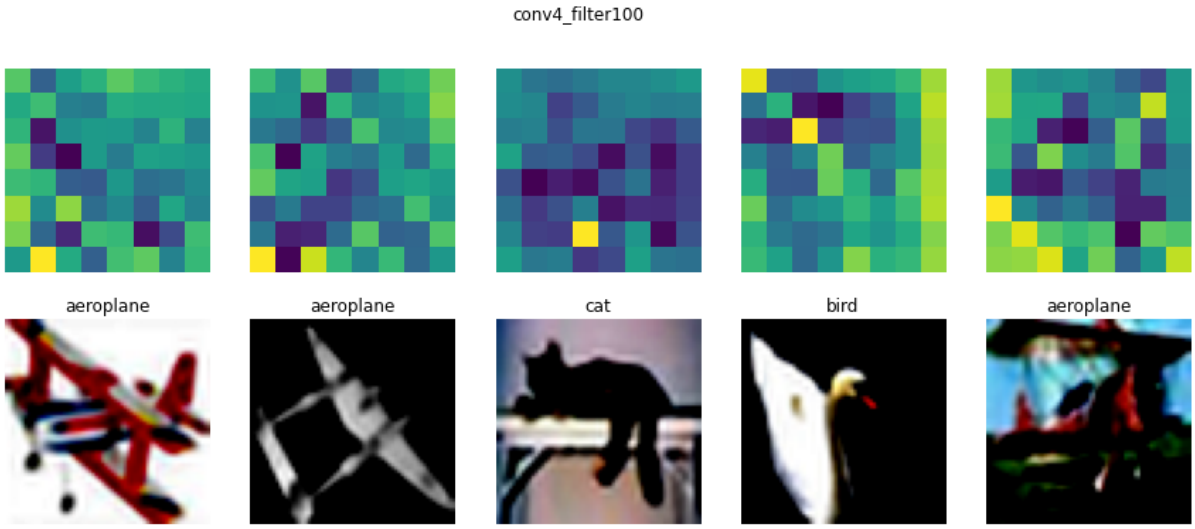
Output of 63rd filter of 2nd conv layer



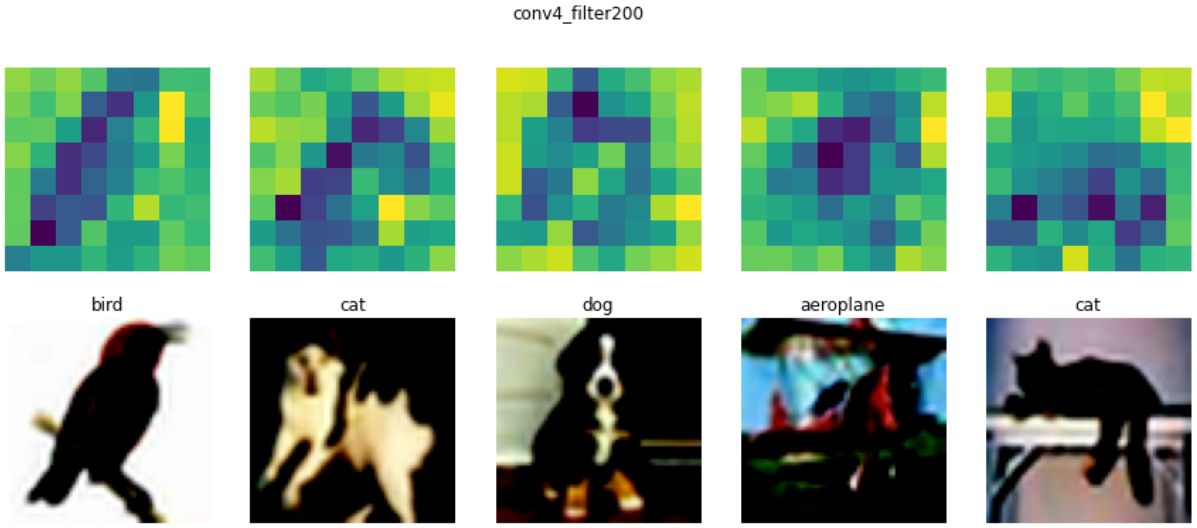
Output of 5th filter of 3rd conv layer



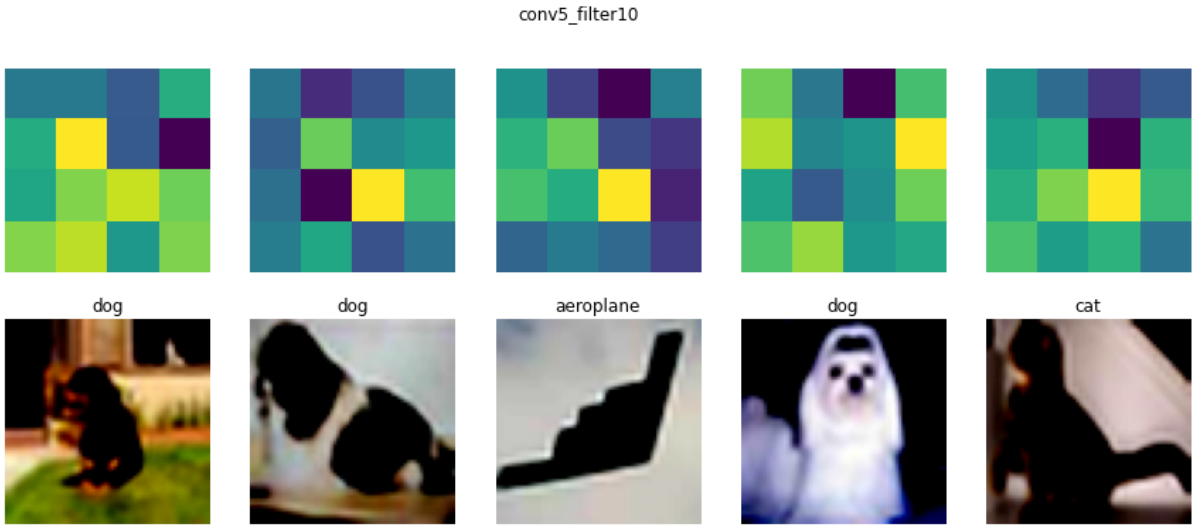
Output of 80th filter of 3rd conv layer



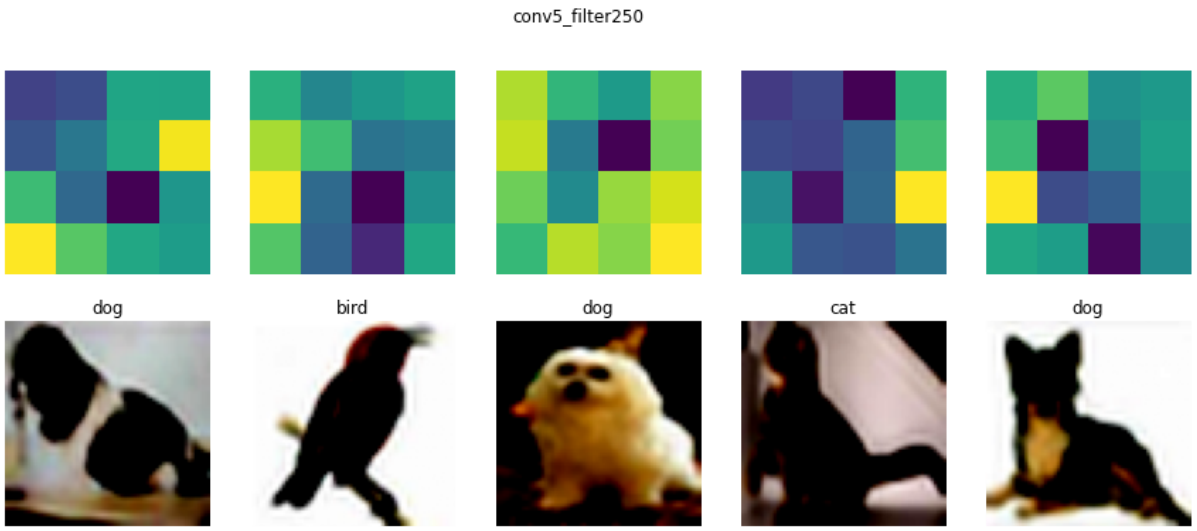
Output of 100th filter of 4rd conv layer



Output of 200th filter of 4rd conv layer



Output of 10th filter of 5th conv layer



Output of 250th filter of 5th conv layer

Conclusion

Here is an example of what a convolutional network might be looking for, layer by layer.

Layer 1: simple edges at various orientations

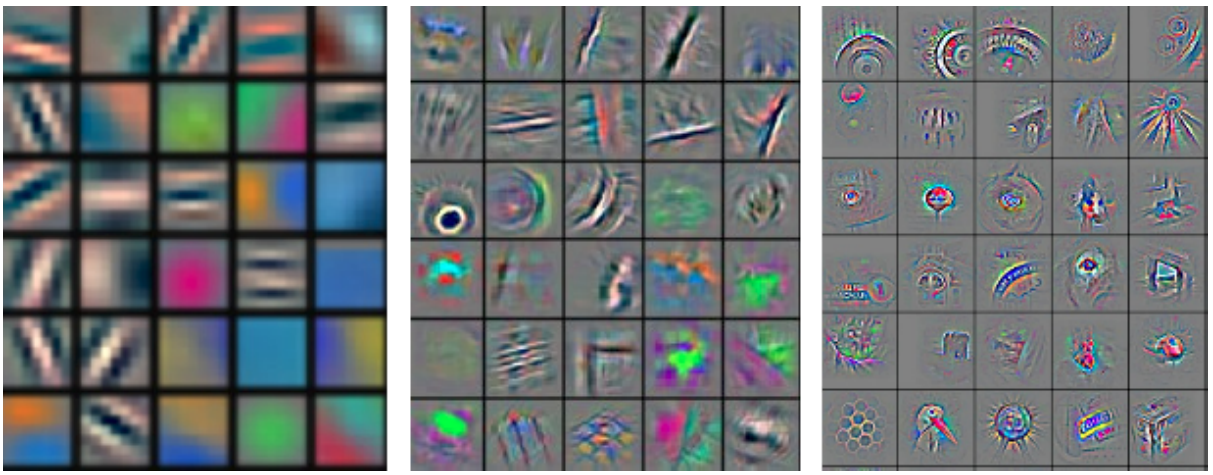
Layer 2: combinations of simple edges that form more complex edges and textures such as rounded edges or multiple edges touching

Layer 3: combinations of complex edges that form parts of objects, such as circles or grid like patterns

Layer 4: combinations of object parts that form whole objects, such as faces, cars, or trees

Each layer has a much wider scope of different things it can look for. The first layer doesn't have a wide variety of things to look for because it is so general. There are vertical edges, horizontal edges, diagonal edges, and some angles in between. But the final layer has a huge variety of things it could be looking for, so a larger number of filters is beneficial.

This image may also make it clearer. A technique is used to visualise the learned filters of the first, 2nd, and 3rd layers.



From the filter identification experiment we can observe that the model at the earlier layers tends to focus on simple features like edges, corners , places where there are sharp gradient changes etc. However as you go into the deeper layer more information is being dimension-ally reduced down and represent in a lower form.The features here can be more general like eyes, body parts, shapes, etc. which are more complex and not easily representable.

2.2.2 Filter Modification

Comparison

Accuracy before switching the weights off

```
Accuracy of the network on the test images: 80.280000 %

Accuracy of  aeroplane : 85.600000 %
Accuracy of       bird : 80.800000 %
Accuracy of        cat : 64.000000 %
Accuracy of        dog : 82.000000 %
Accuracy of       ship : 88.800000 %
```

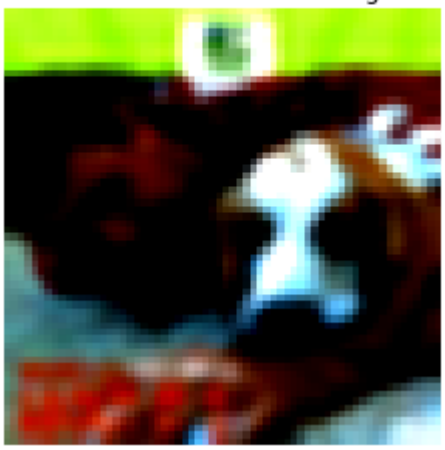
Accuracy after switching the weights off

```
Accuracy of the network on the test images: 78.6 %
Accuracy of  aeroplane : 84.800000 %
Accuracy of       bird : 81.200000 %
Accuracy of        cat : 62.600000 %
Accuracy of        dog : 80.800000 %
```

```
Accuracy of        ship : 87.000000 %
```

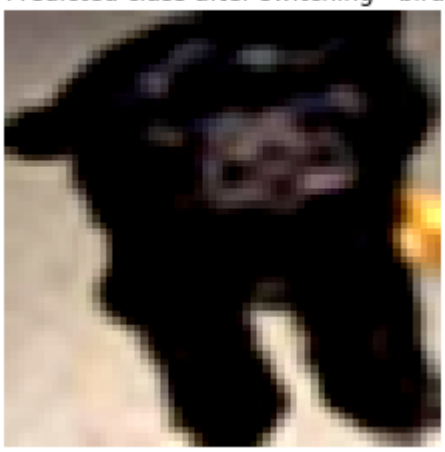
Number of images that were misclassified by model(after switching the weights off and were initially correctly classified when the weights were not switched off are 92

True class-dog
predicted class before switching - dog
Predicted class after switching -cat

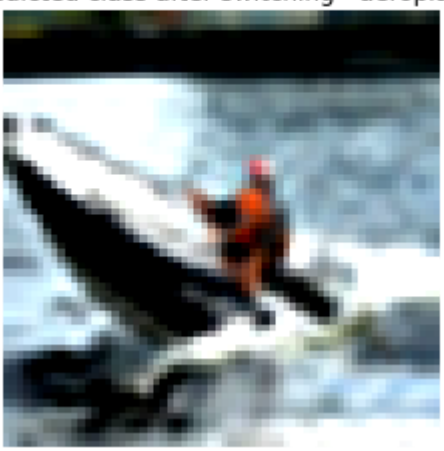


True class-ship
predicted class before switching - ship
Predicted class after switching -aeroplane

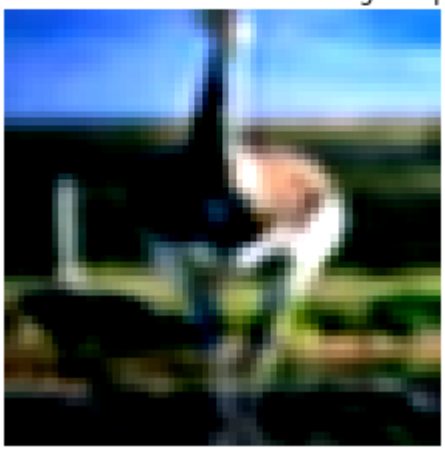
True class-dog
predicted class before switching - dog
Predicted class after switching -bird



True class-ship
predicted class before switching - ship
Predicted class after switching -aeroplane



True class-bird
predicted class before switching - bird
Predicted class after switching -ship



True class-aeroplane
predicted class before switching - aeroplane
Predicted class after switching -bird

True class-aeroplane
predicted class before switching - aeroplane
Predicted class after switching -ship

True class-bird
predicted class before switching - bird
Predicted class after switching -aeroplane

Conclusion

Before switching the weights ,the model mis-classified aeroplane as ship and vice-versa and but now the model is not able to distinguish between the bird,aeroplane and ship (after switching the filters).
We can conclude that the filters that were extracting the common or similar features from bird , aeroplane and ship are switched off .
Switching off the filters also helps to further understand and infer what the model has really learnt and what features it is focussing on.