Autoencoders – Deep Networks Visualizations

I had begun this semester learning about Neural Networks in general and its applications. I later dived into autoencoders and was regenerating images given a set of few data points. My report here includes an analysis of the filters generated through the network and how to interpret them in order to improve their performance.

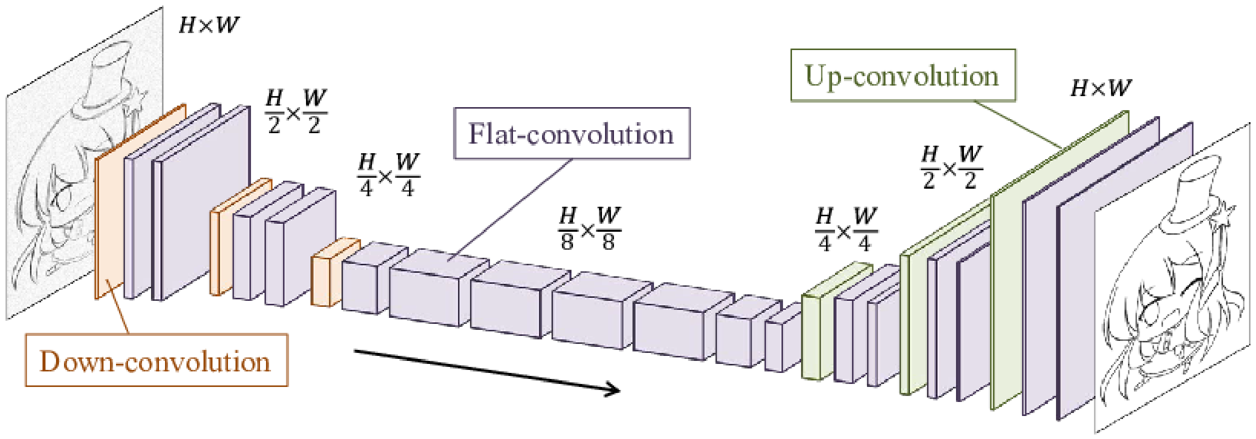

$H \times W$
$\frac{H}{2} \times \frac{W}{2}$
$\frac{H}{4} \times \frac{W}{4}$
Flat-convolution
$\frac{H}{8} \times \frac{W}{8}$
Up-convolution
$\frac{H}{4} \times \frac{W}{4}$
$\frac{H}{2} \times \frac{W}{2}$
$H \times W$
Down-convolution

Fig 1. Typical Autoencoder

Using the cifar10 dataset that has around 50k train images and 10k test images, I build an autoencoder. This autoencoder down-sampled the data and later up-sampled it with fewer data points (pixels) to generate the entire picture. The network was build similar to a convolutional neural network. The first half of the neural network constituted of 3 layers that brought down the picture size from 32 X 32 to 5 X 5. Later, the encoded 2d matrices were upsampled to its original size of 32 X 32. The decoded layer is then compared to a test image, for performance. To understand what the network really understands, I passed a test image through each layer and created visualizations to try and understand what each layer is learning. And from that gained some insights to improve performance. Below, I have summarized what the filters have learnt throughout the network.

Filter Learning:

The first activation layer, learns the shapes and patterns of objects on an intricate level. It can be visualized as below.

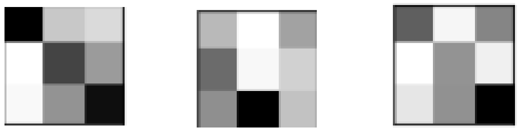


Fig 2. a) Filter looking for diagonal dark boundaries, b) Filter looking for vertical light boundaries, c) filter looking for vertical grayish boundaries

Now it gets more interesting when we go to layer 2. Here, it wasn't obvious what was happening. It's learning a much more complex set of patterns than the first layer edges and you can see its combining them with in meaningful ways like parallel lines, curves, circles, grading patterns; a huge variety of different structure is present, and we are already in the second layer. You can see it covers a much larger space of the image because it's gone through the first set of convolutions, then there is a pooling layer in between the second layer convolution, so it has a much broader scope of what we can see in the image. Since the images are of size 6X6 it isn't easy to interpret to the naked eye.

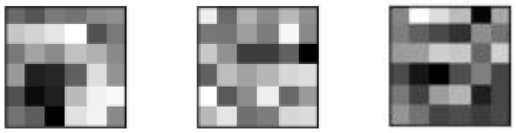


Fig 3. Learning a complex set of patterns

Moving to the third layer, it's getting more and more complex. It has now gone through 3 convolutional layers and pooling layers. You can see it has more object patterns than before, showing that the network has been learning well from a small baseline to a more complex structure through each layer. Again, since it is just a 9X9 image, it is hard to interpret these images by the naked eye, but in the next section we will look into what is the purpose of looking at these images and how it will help improve our performance.

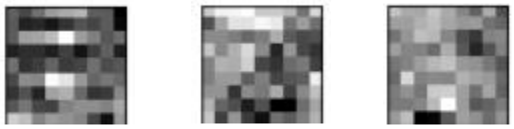


Fig 4. Learning a more complex set of patterns

Improving Performance:

How can we use these visualizations to our benefit?

We can actually use visualizations to improve the performance of our network. One of the common strategies is to visualize the weights. These are usually most interpretable on the first CONV layer which is looking directly at the raw pixel data, but it is possible to also show the filter weights deeper in the network. The weights are useful to visualize because well-trained networks usually display nice and smooth filters without any noisy patterns. Noisy patterns can be an indicator of a network that hasn't been trained for long enough, or possibly a very low regularization strength that may have led to overfitting. Here are a few points for consideration.

1) Normalization

Below you can see a test image passed through a single convolutional layer. If there was no normalization, I was getting an image which was a really bright image and completely darkened images for the rest of my filters and simultaneously my losses were also high for that network. By visualizing them, I see that there is a huge disparity between the normalization of the filters in this layer, and they need some form of normalizing.

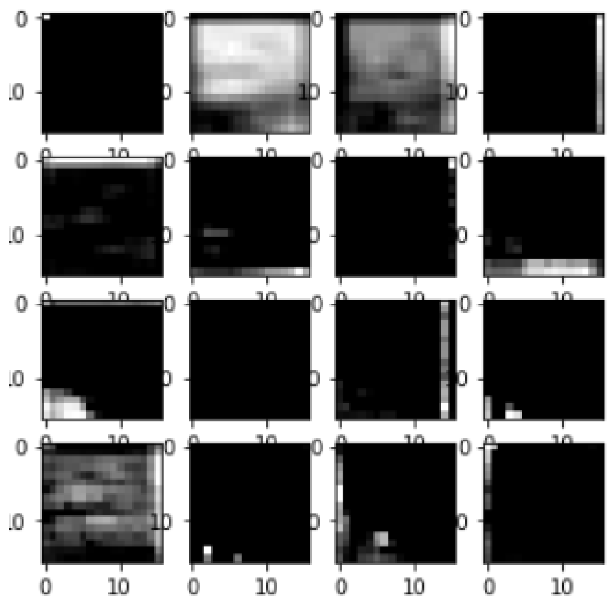


Fig 5. Needs better normalization

2) Artifacts

After normalizing, besides looking at just the structure, it might be possible to encounter artifacts, some filters maybe just artifacts that need to be removed. So, you do need to do some kind of renormalization in order to fix this problem and help all the features learn better.

By artifacts, I mean randomly generated patterns, or dead filters. These are generated due to large size in filters. For eg, using a 9 X 9 filter for a 32 X 32 image in its first layer won't capture the images on a lower level (i.e. lines, curves, circles), but capture a large content of the image when you look at the 9 X 9 patches.

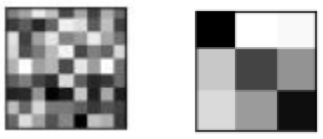


Fig 6. Using a 9X9 filter vs 3X3 filter for its first convolutional layer

From the figure above, it is clear that the smaller size filter is trying to capture the lower level intricacies of an image, which is required for the first convolutional layer. Subsequently it was proved by the lower validation loss I got at the end of training the network. (First Conv. Layer:- 3X3: val_loss =

0.5802, 9X9:val_loss = 0.9090). This would prove to be inefficient for networks to learn. This would be an indication to lower your filter size at the beginning of your network.

3) Blocking artifacts

Another set of filters one may encounter, is the blocking artifacts. This a key finding for a CNN as it blocks/ losses a lot of information here.

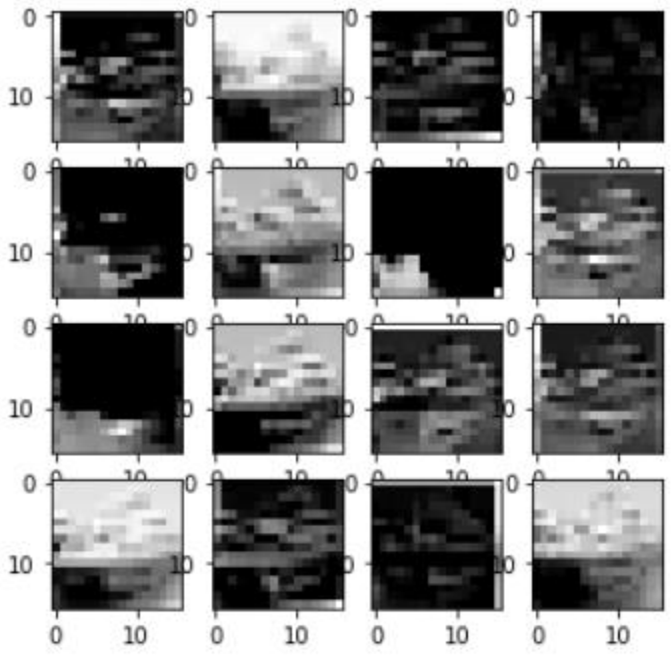


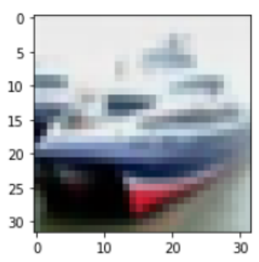
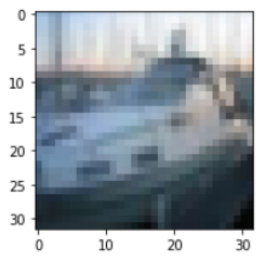
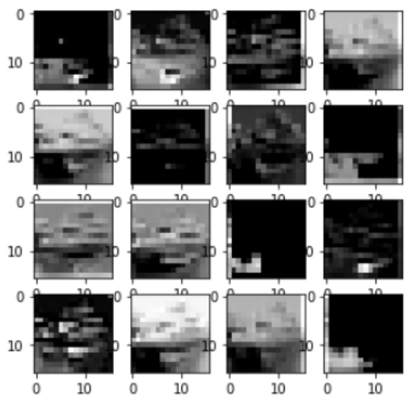
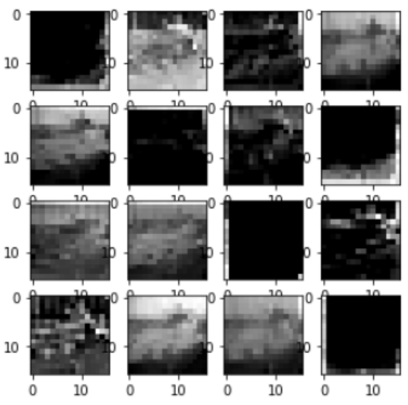
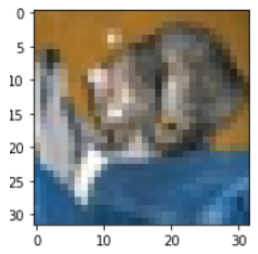
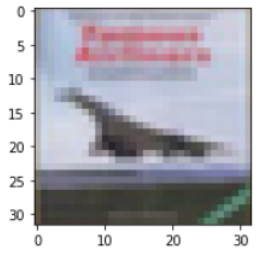
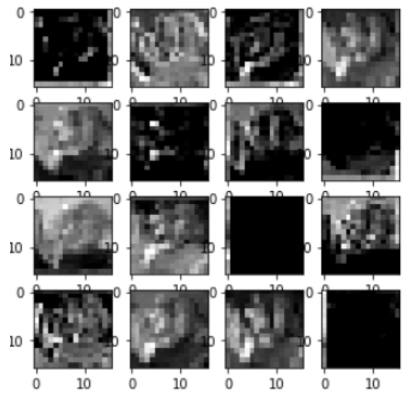
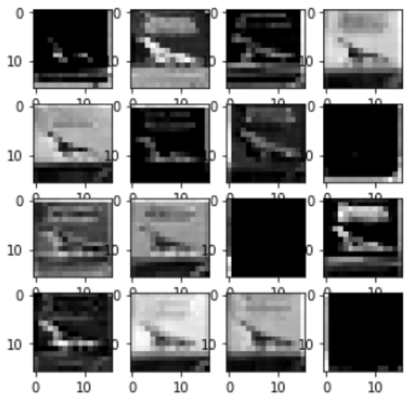
Fig 7. Top right (row – 1, col - 4): blocking artifact

To overcome these tasks, one can go back to renormalization by placing a threshold on the values of the filters based on their rmse. And that makes all of them compete evenly with each other. Another suggested method is to reduce the stride in their convolutional neural network. It would make that particular layer much more flexible and learn much more interesting content.

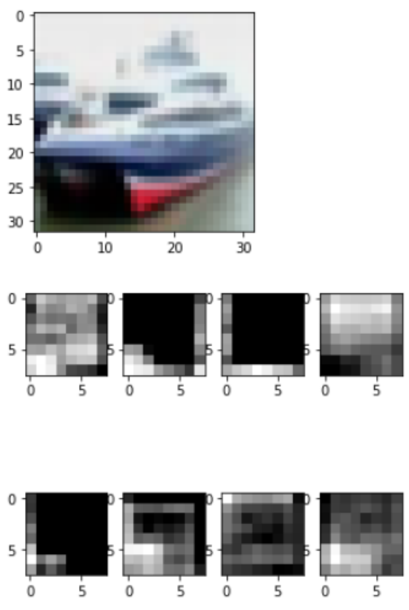
And to top it all off, in case one finds filters/activations that aren't helpful of any sort, they should try reducing their filter size as it would be simpler for their network to learn patterns and simultaneously a huge variety of content. Below, I have shown 4 test images passed through each layer.

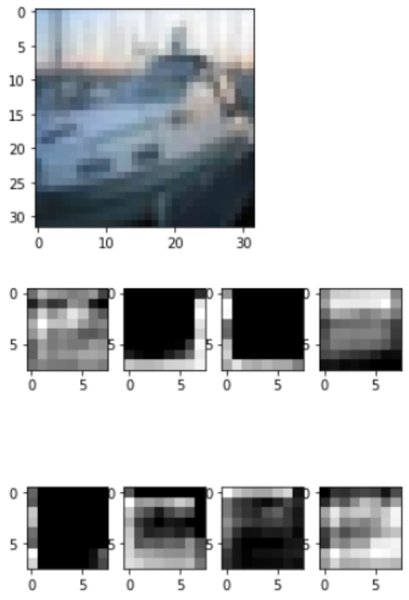
Passing different test images through each layer:

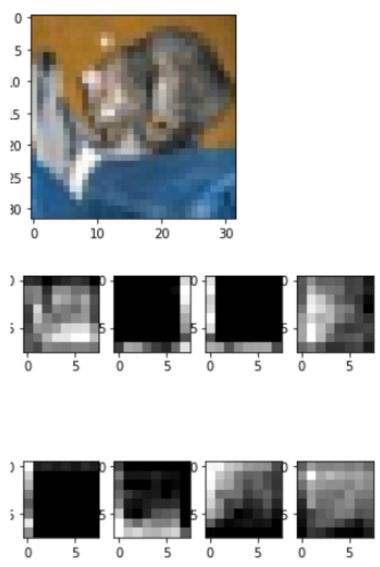
Layer 1 –

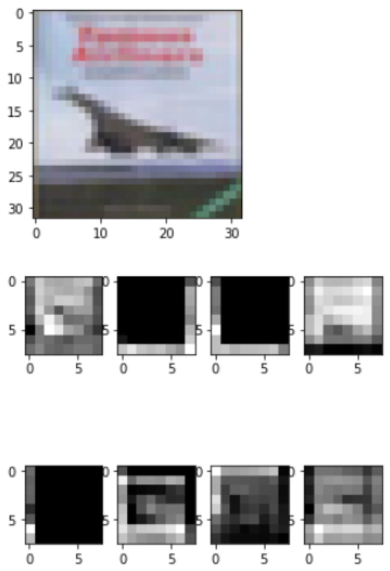









Layer 2 –

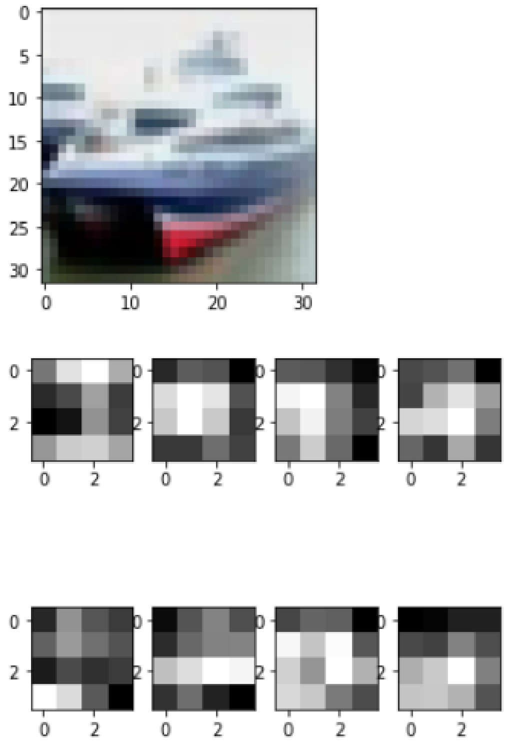


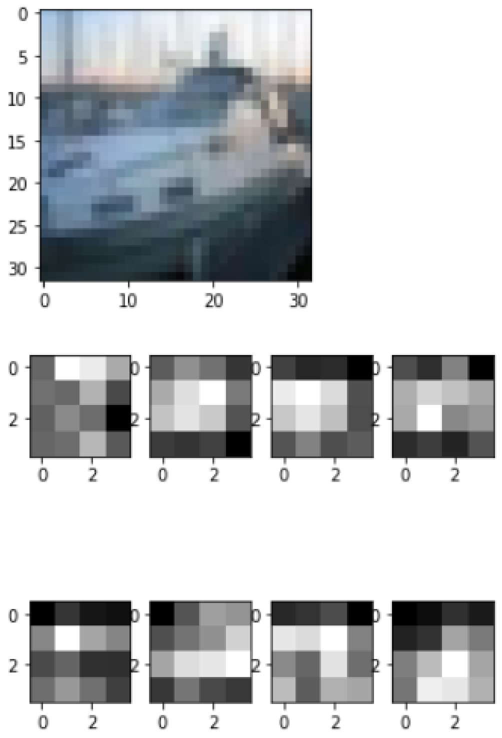


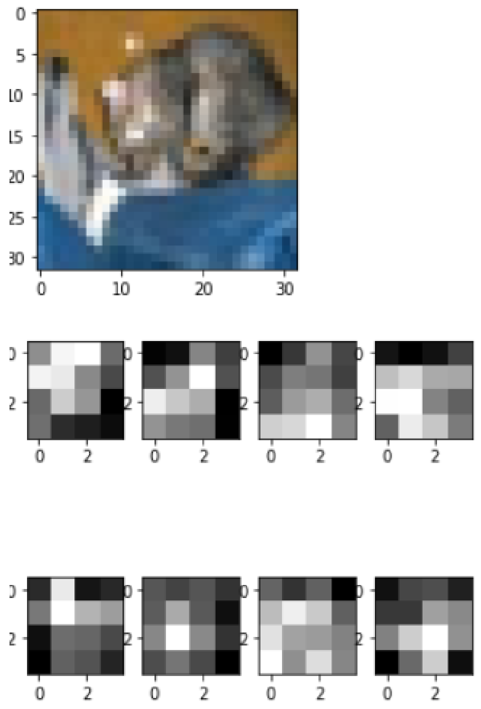


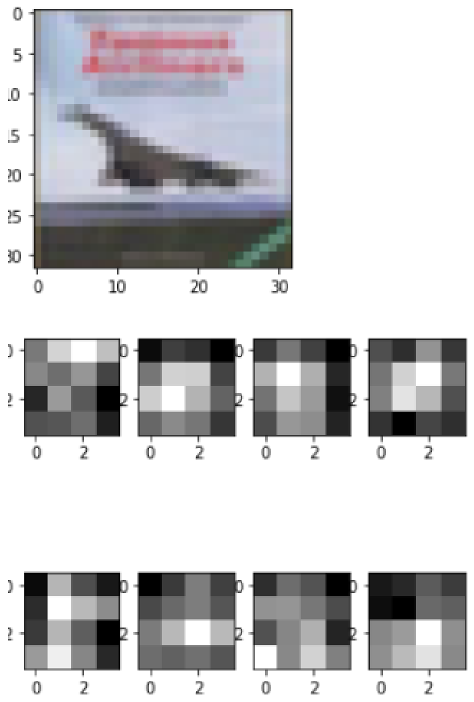


Layer 3 -

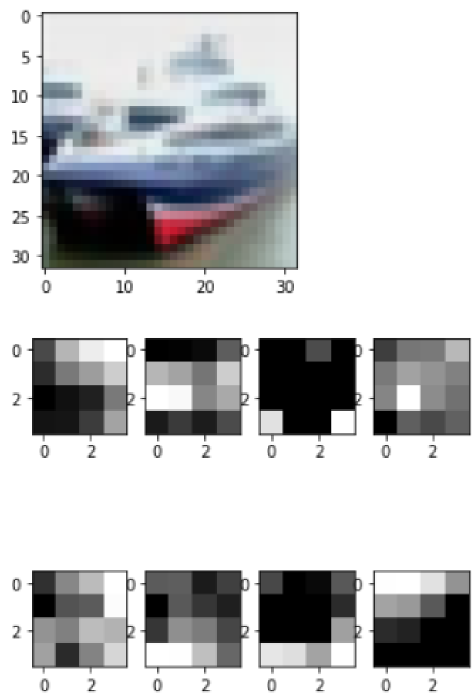


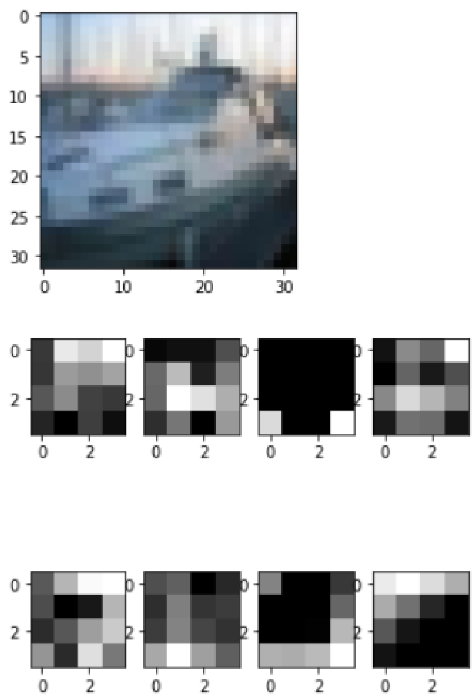


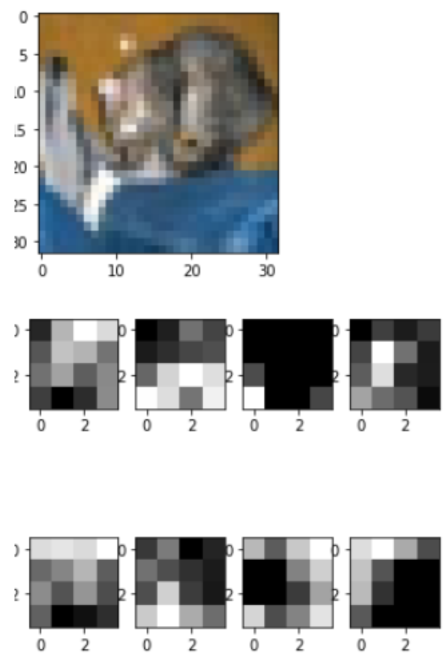


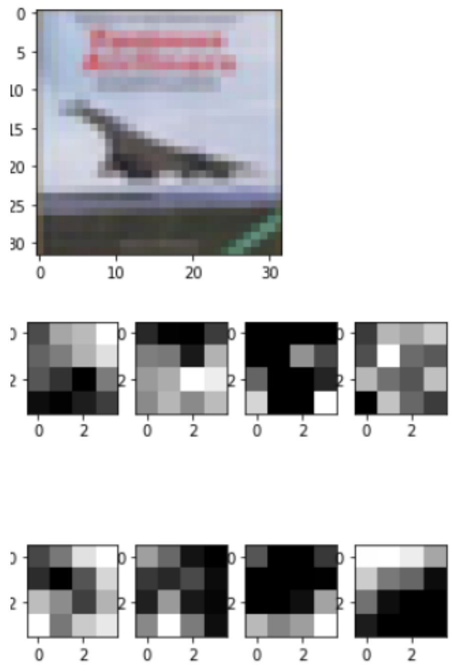


Layer 4 – Upscaling

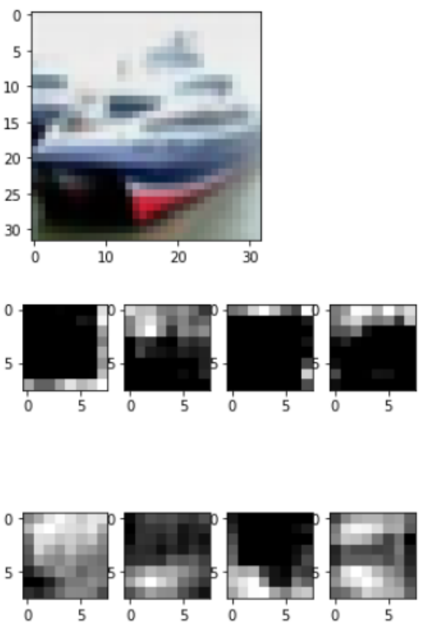


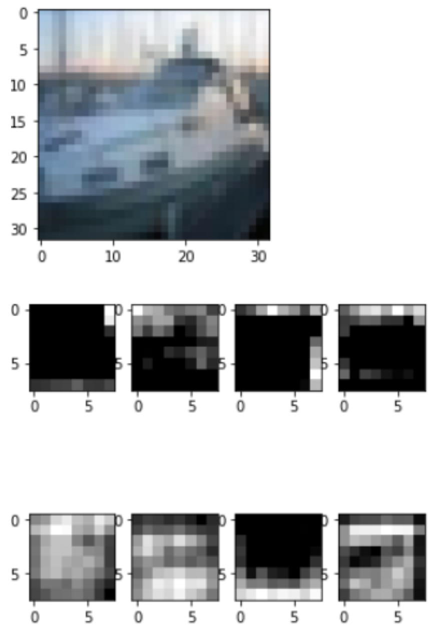


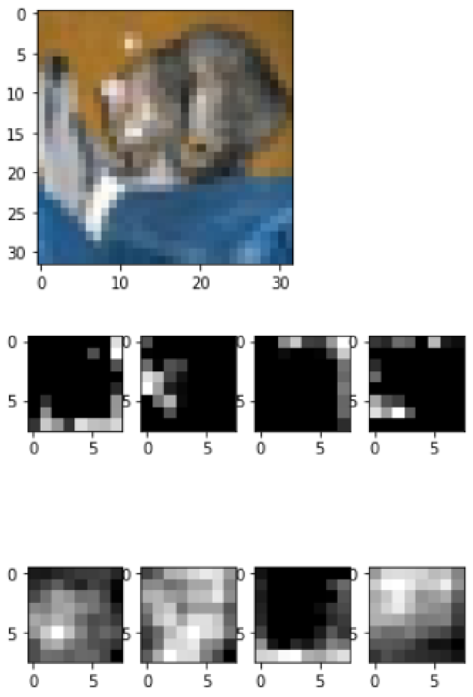


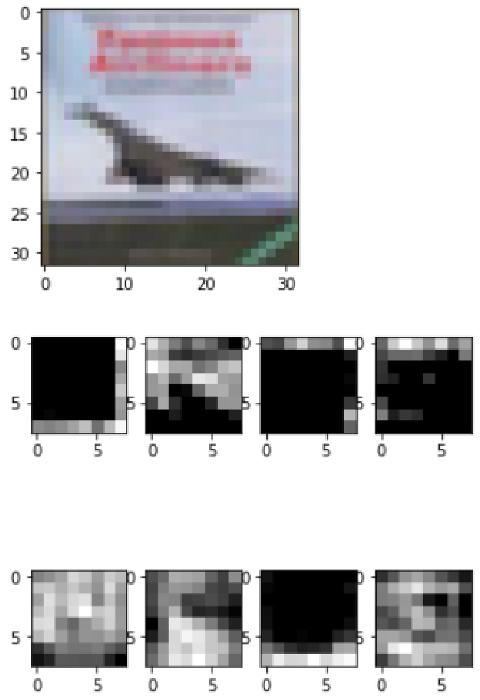


Layer 5 –

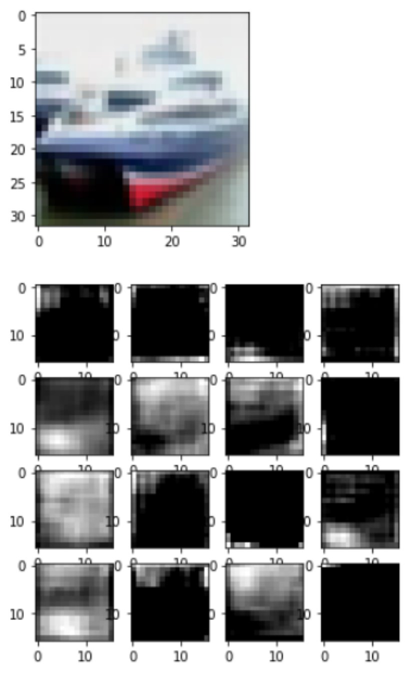


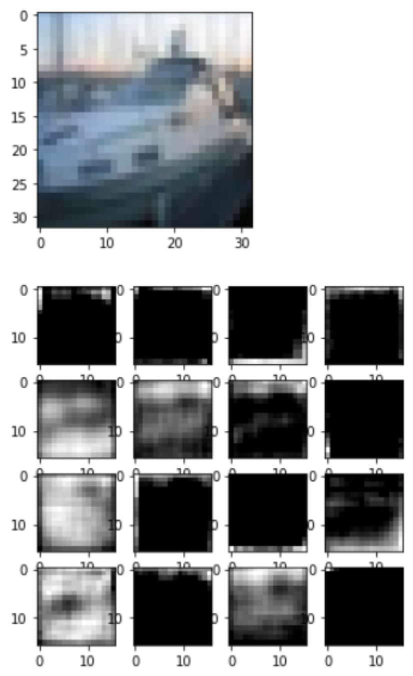


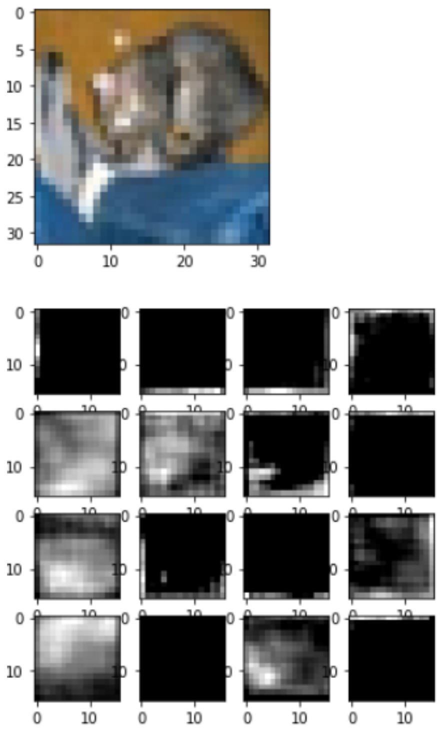


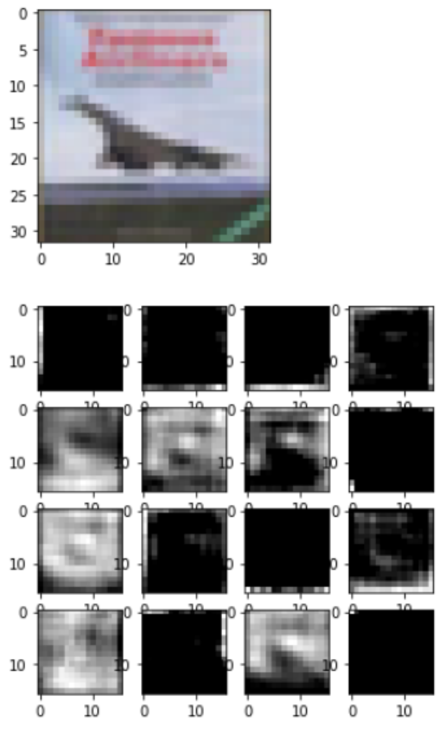


Layer 6 –









Decoded Layer –

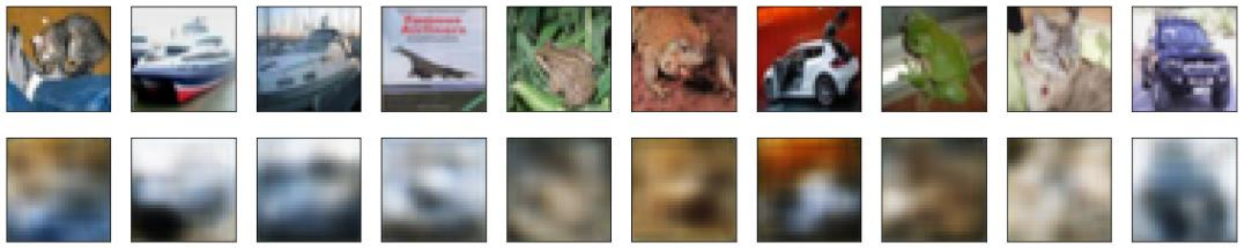


Fig. 8 Final Output layer