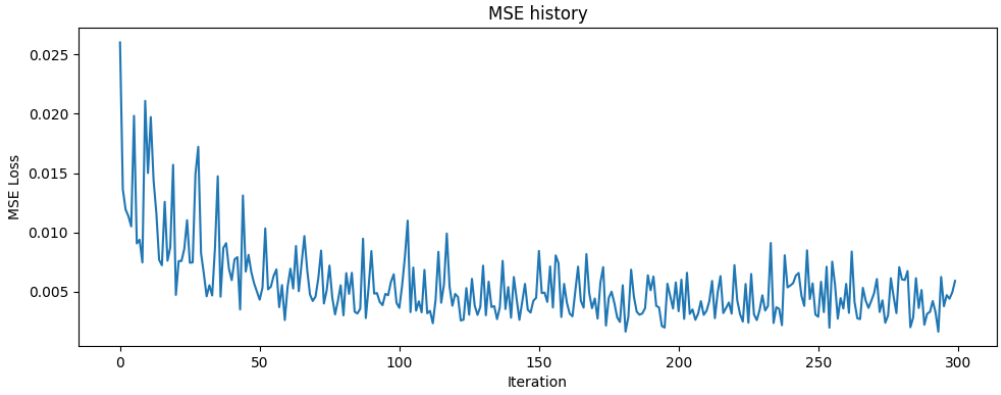
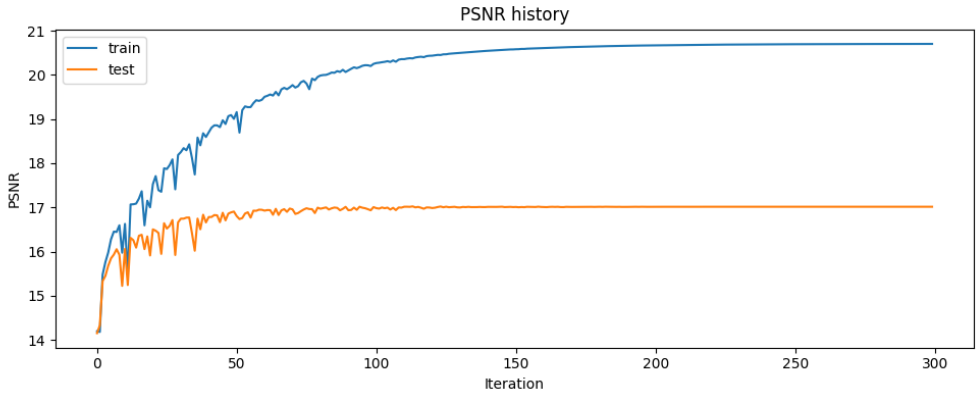
Assignment 2 Report

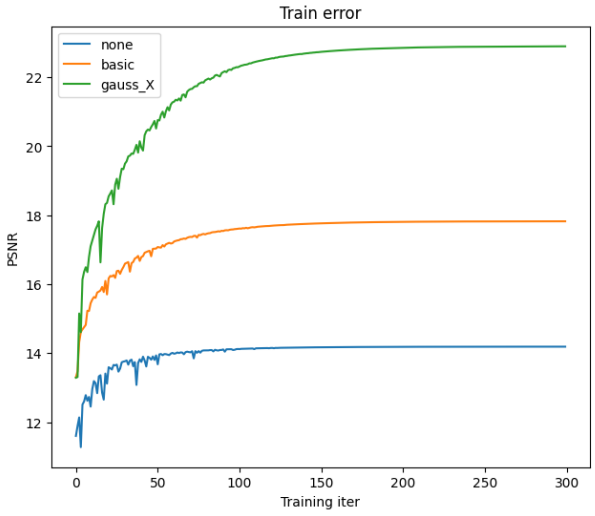
Name(s): Girish Madhavan Venkataramani
NetID(s): gmv4

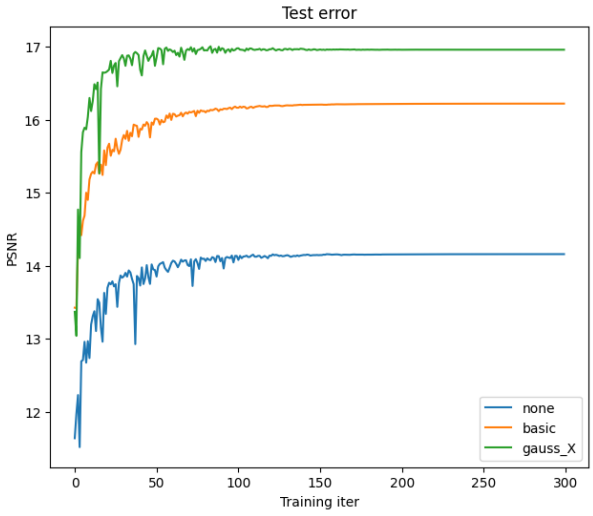
Part 1: Low resolution example

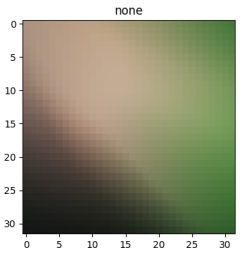


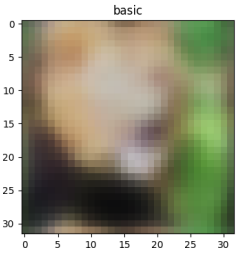


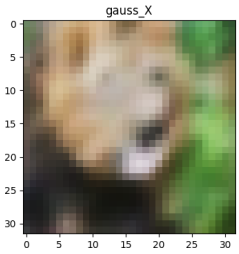
Learning progress for Gaussian Fourier Feature mapping

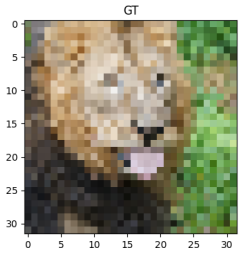




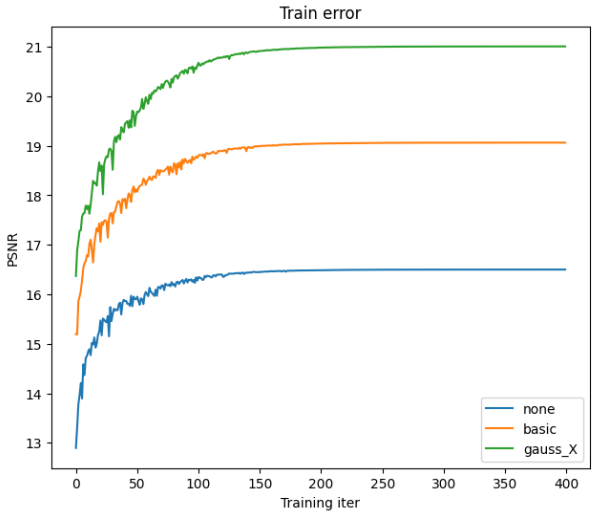


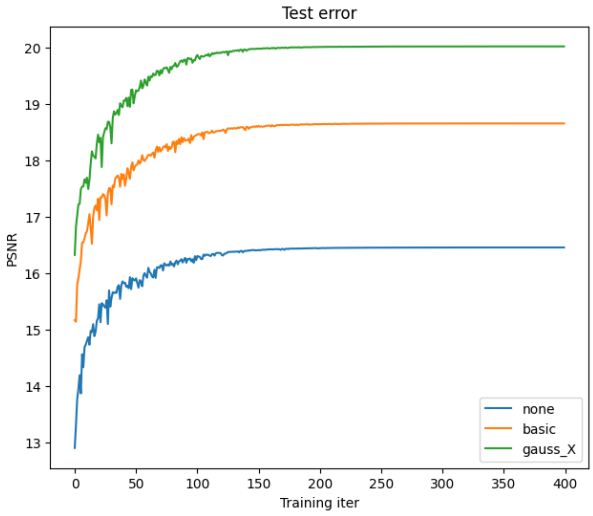


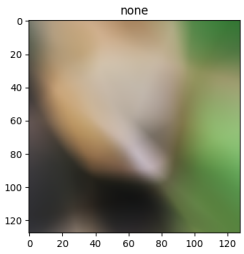


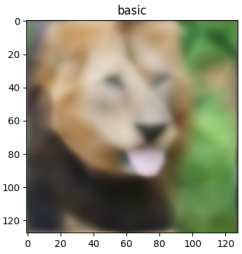


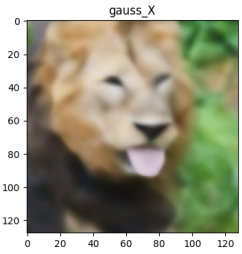
Part 2: High resolution example

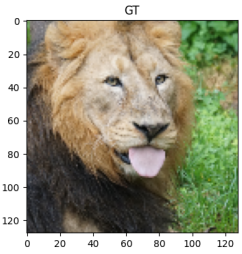




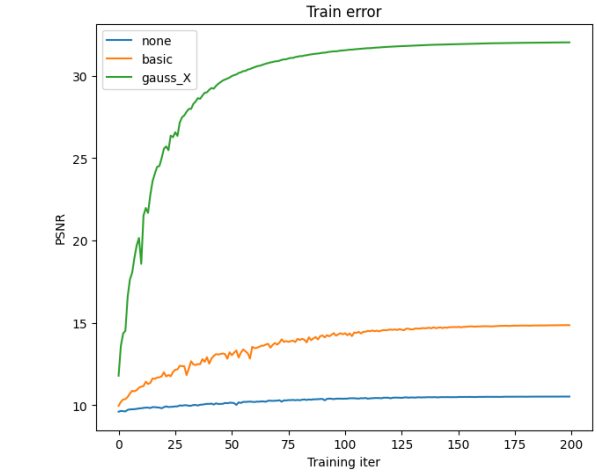


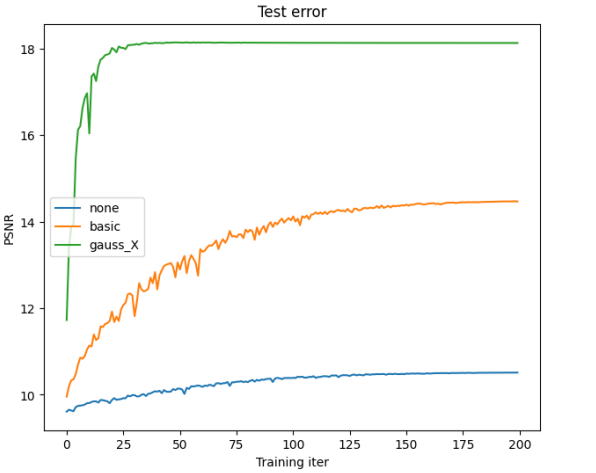


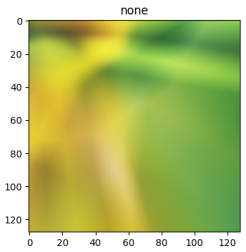


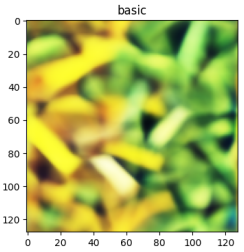


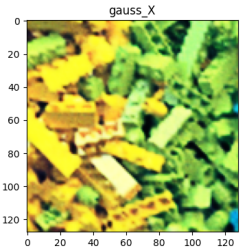
Part 3: High resolution (image of your choice)

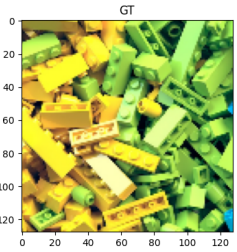


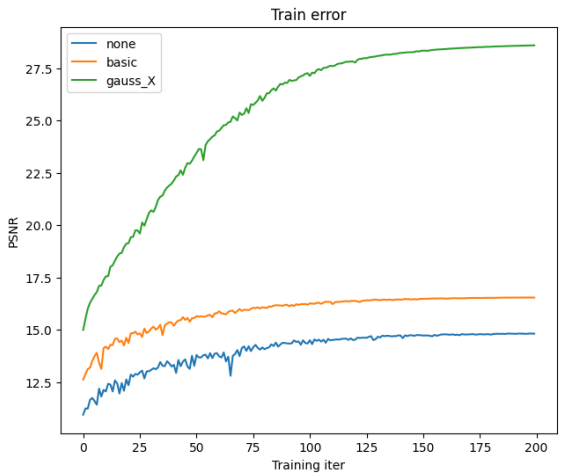


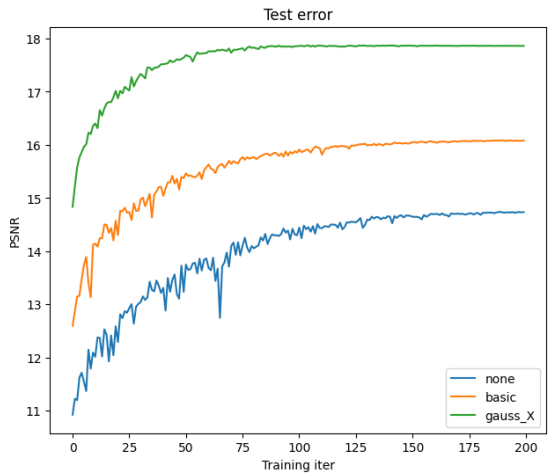


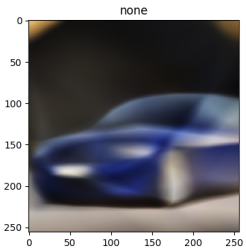


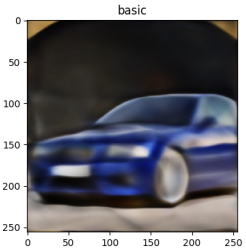


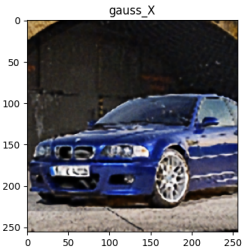


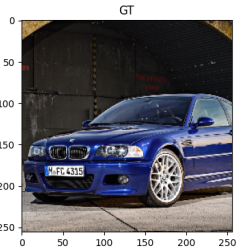












Part 4: Discussion

Hyperparameter tuning :

- Changing batch size in mini batch mode : single batch mode of training gave lower psnr and a very noisy mse, while full batch gave a smooth convergence of mse, it struggle to improve its psnr. Mini batches of sizes 16, 32, and 64 were tried for low and high resolution images, out of the three a batch size of 32 gave the best psnr values. This was the most impactful hyperparameter of all
- Learning rate : with SGD, a low learning rate like 0.1 or even 1 incapable of learning the rapidly changing features of the image, furthermore the addition of a learning rate decay term completely prevented the model from giving any useful results. This issue was fixed when increasing the learning rate to high value like 60 or more, going more than 60 showed no improvement in the test psnr values. This could be due to overfitting during training, moreover higher the learning rate more noisy the psnr values were in the initial training process which eventually died down
- Frequencies in the B matrix : values like 32, 64, 128 were tried and no significant changes were observed for both cases, this could be because, we are restricting the standard deviation of normal distribution used in generating B matrix as 1, effectively minimizing the chances of capturing sharp details in the image. This doesn't affect the low resolution image much because it doesn't contain any sharp details but the high resolution image improves significantly with higher standard deviation values

Effect of feature mapping function [$\gamma(v)$] :

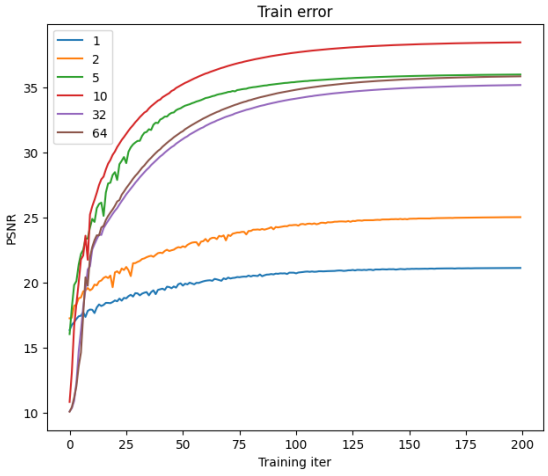
- No mapping : struggled in all tasks and gave very low psnr values and high mse
- Basic mapping : significant improvement over not having any mapping but still doesn't perform well in scenarios when higher frequencies are needed, as this analogous to fourier feature mapping with only having one frequency
- Gaussian Fourier Feature mapping : significant improvement over basic mapping as now more frequencies can be separated, performs very well with increased std of normal distribution

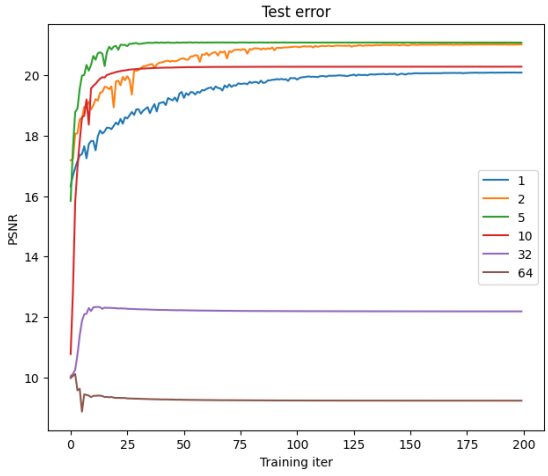
One interesting observation was the psnr values increased with increasing the resolution of the image and for every resolution, there seems to some kind of a barrier to the psnr values crossing this seems to be almost impossible. For example for the high resolution cases, going anymore than 20 was difficult.

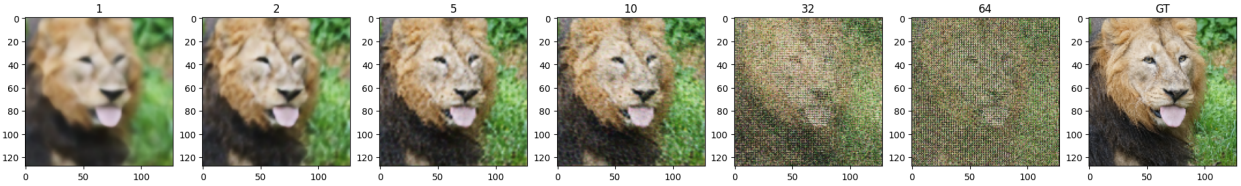
Part 3 : in this setting the B matrix is of dimension [128 x 2] and uses a standard deviation of 5, as going above 5 produces more noise in the output. Two types of images were trained on, the lego blocks was a lower resolution of size 128x128 and the bmw was a higher resolution of 256x256. From both images we can see the barrier as mentioned previously and a closer look at the rendered images show, continuous regions like the surface of the car or the sides of a lego block were reproduced much better than the edges or curves in the case of gaussian mapping. Without any mapping, the model is unable to do anything in cases when there are too many objects in the image like the lego blocks, but in the case of bmw, it was able to separate it out from its background. We can also observe the scaling of performance wrt increasing image resolution, although this effect is not much pronounced with the gaussian mapping, its much more evident in the case of basic and no mapping.

Part 5: Extra Credit (optional)

5.1.1 : Changing the underlying distribution of B mapping

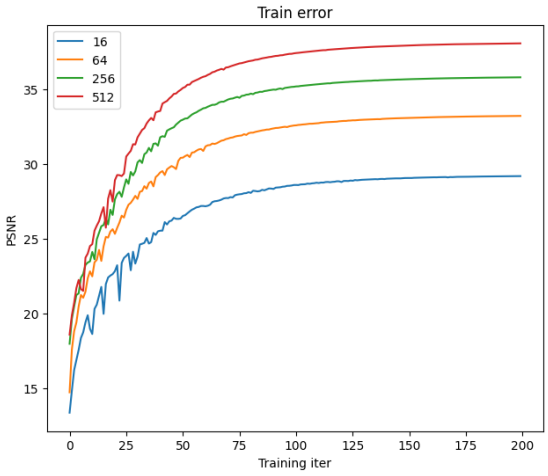


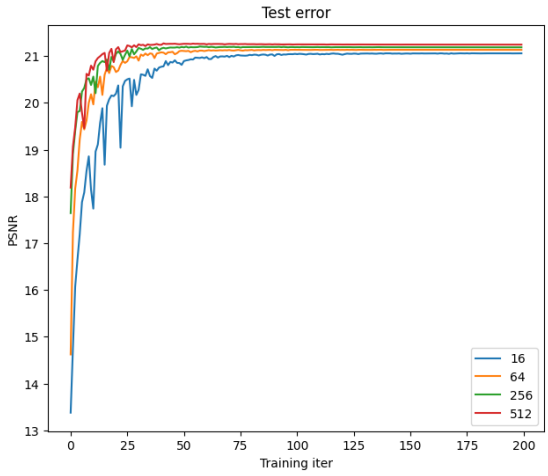


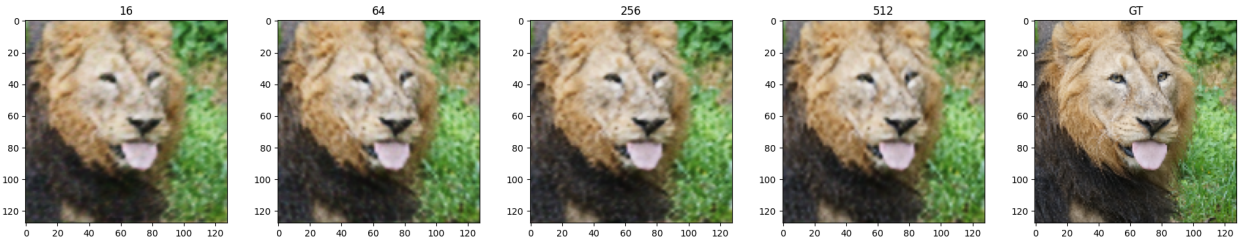


Increasing standard deviation initially improves performance and further increase causes more noise in the output

5.1.2 : Changing the number of frequencies [m] in B mapping

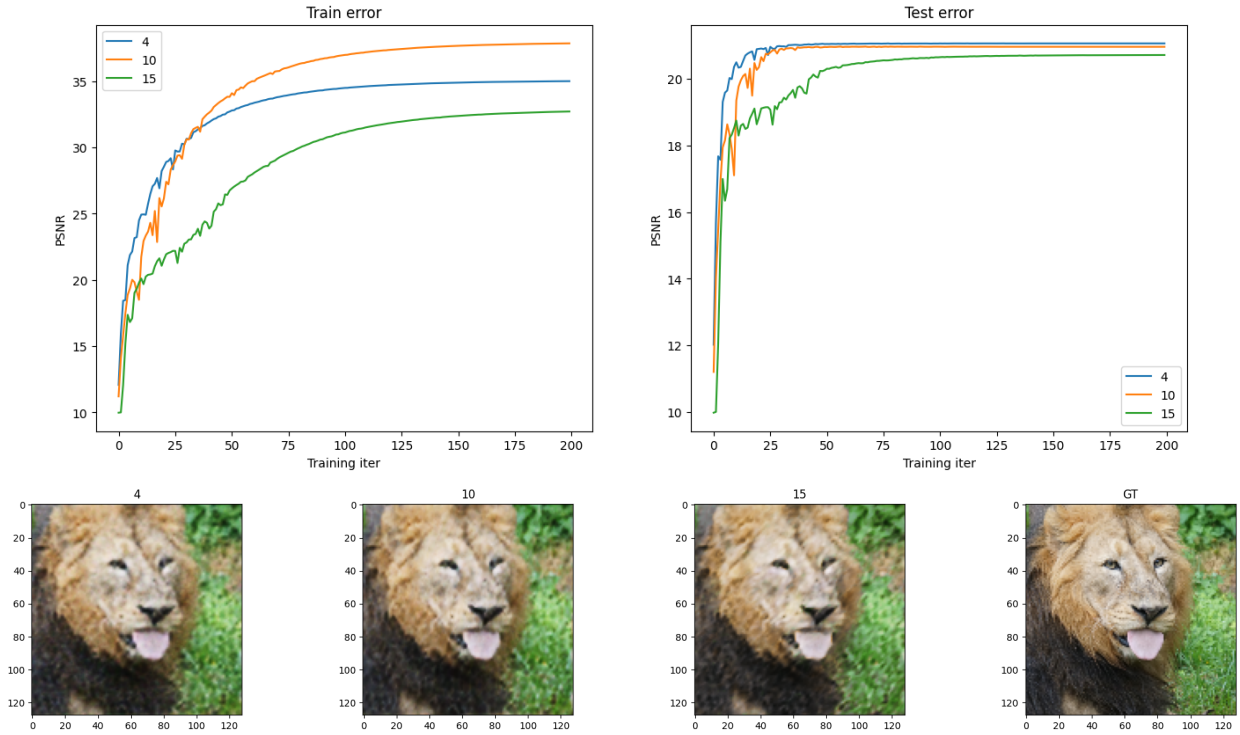






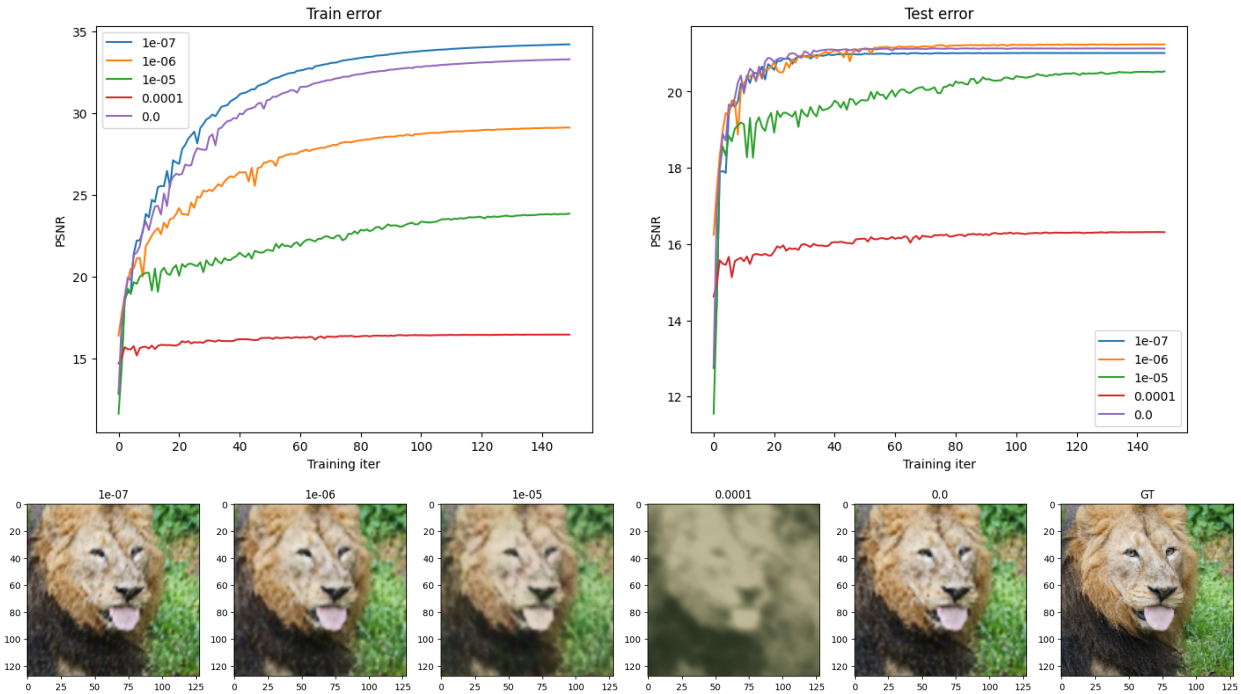
Changing the number of frequencies doesn't affect the test psnr much, although the training error says the performance improves, this could be due to overfitting. A reasonable balance between computation time and test performance would be m = 128

5.2 : Trying a deeper Network



Going any deeper than 4 layers does not provide good psnr values and ends up wasting computation time

5.3 : Regularization !

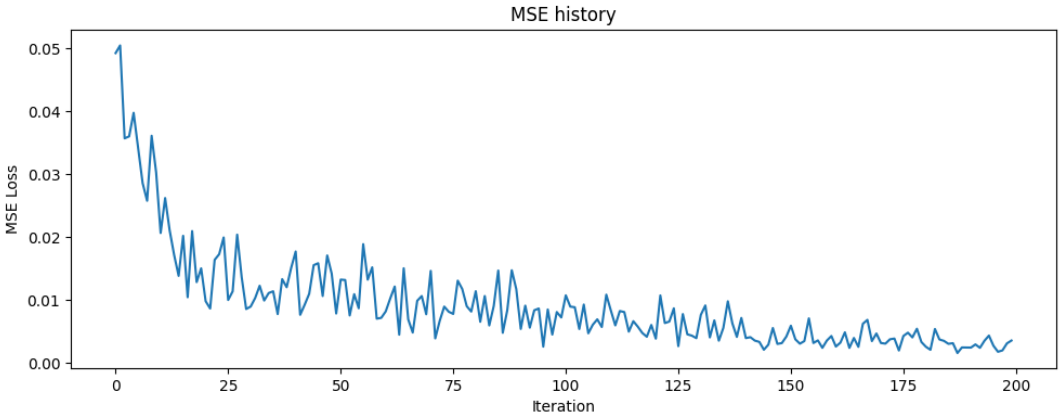


A very low regularization value leads to slightly better performance as compared to without regularization

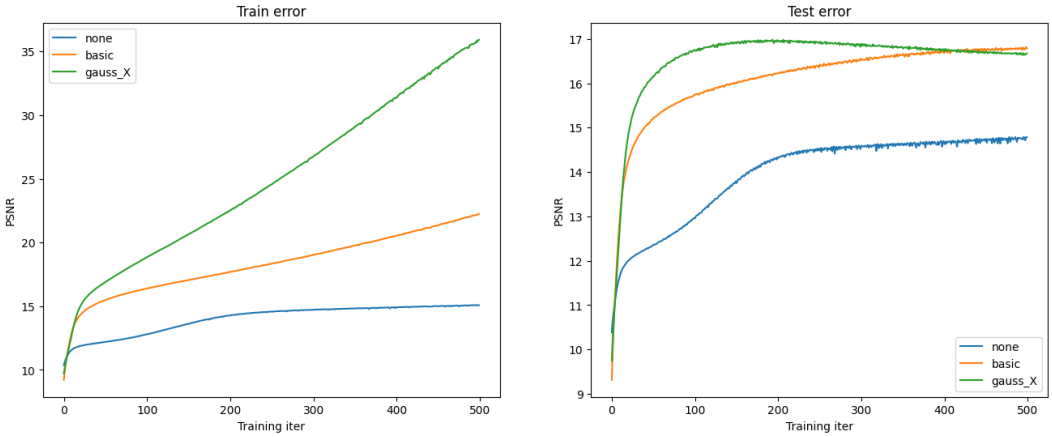
5.4 : Adam optimizer

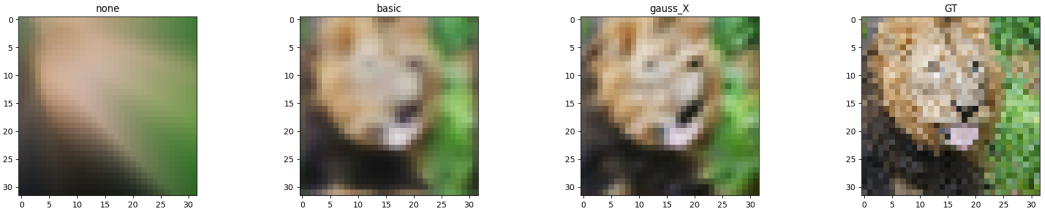
5.4.1 : Low resolution image

Plots and image being generated for Gaussian Fourier Feature Mapping :

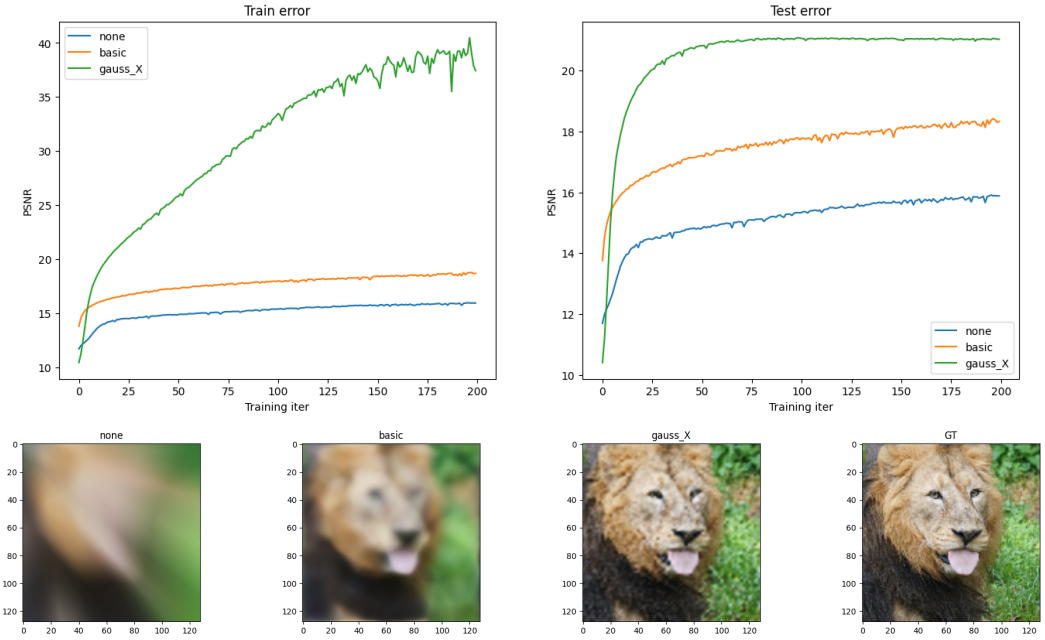


Plots for all cases



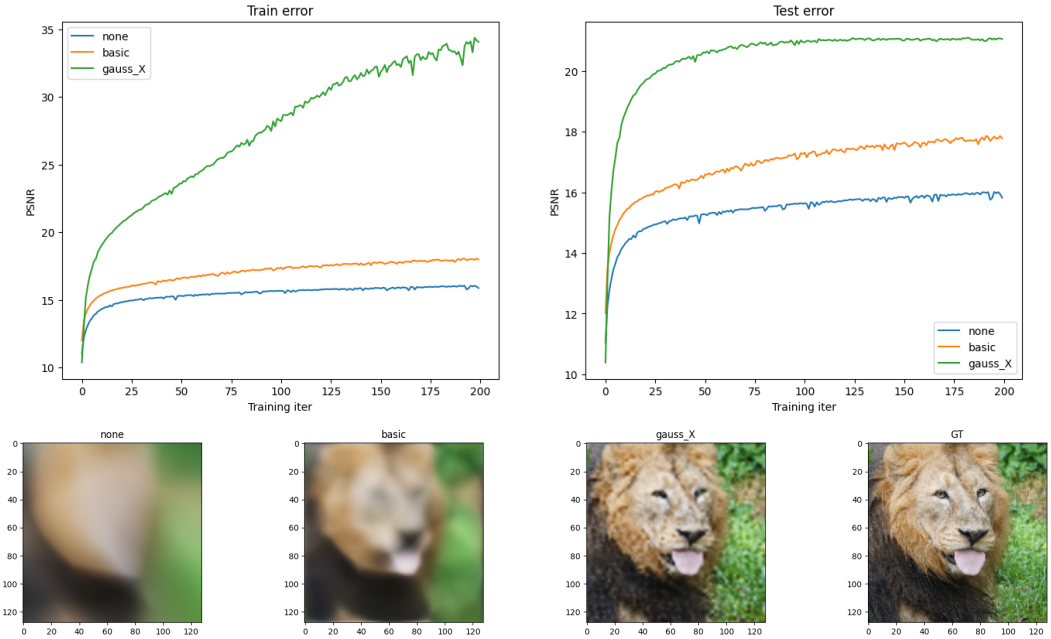


5.4.2 : High resolution image



From low and high resolution images we can see that it trains faster as compared to SGD and the test psnr values are not as jittery as with SGD, this could be due to the adaptive changes in the update process. In the case of low resolution we can see the basic mapping over taking the gaussian mapping in psnr values, possible reason for this because the image doesn't have much detail in it and Adam tends to oscillate with higher epochs.

5.5 : Pytorch



The plots and rendered image looks similar to the implementation using numpy, both the numpy and pytorch were run using the same set of hyperparameters. The computational requirements were much lower as compared to the numpy implementation, taking only about 4 minutes to run all 3 cases, whereas the numpy implementation took around 15 minutes.