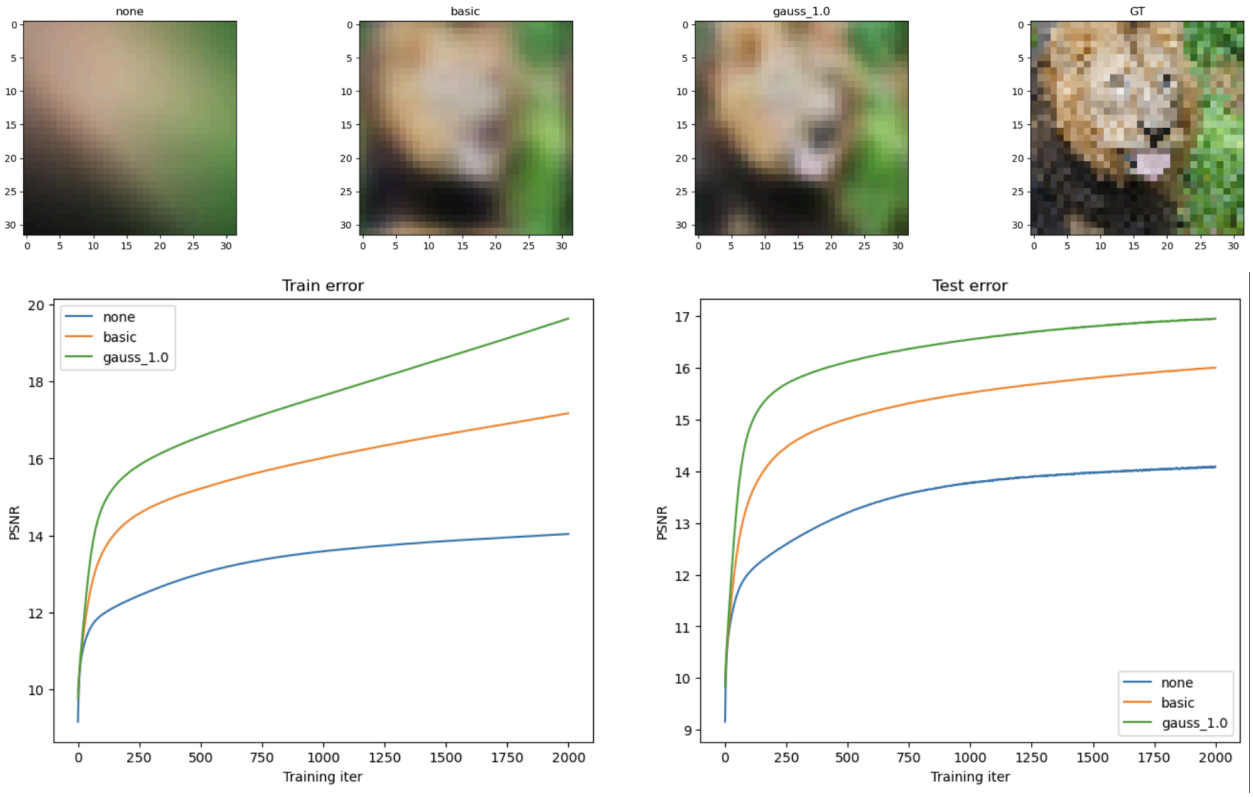
Report

Part 1: Low-resolution example

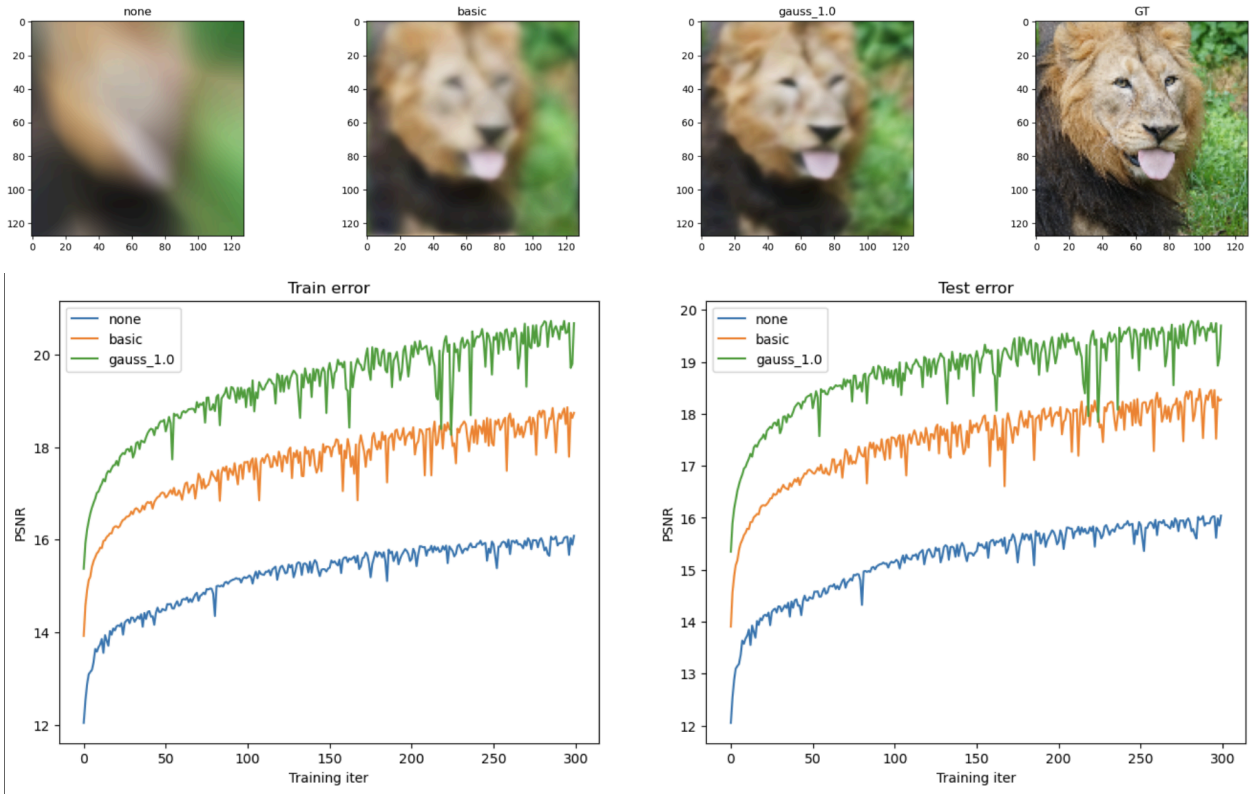
```
num_layers = 4
output_size = 3
hidden_size = 256
epochs = 2000
learning_rate = 0.02
```

For none:
Final Test MSE 0.016735693415374436
Final Test psnr 14.753262930063132



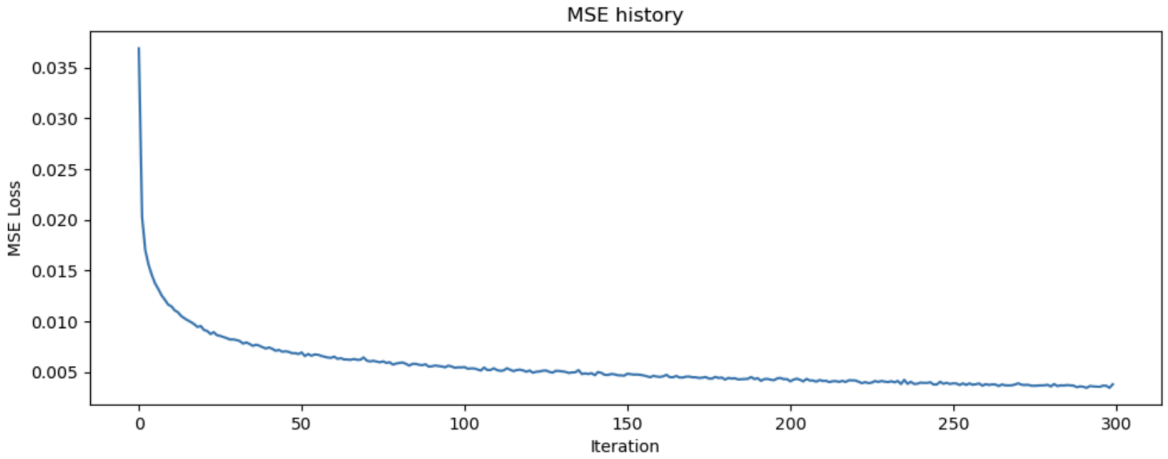
Part 2: High-resolution example

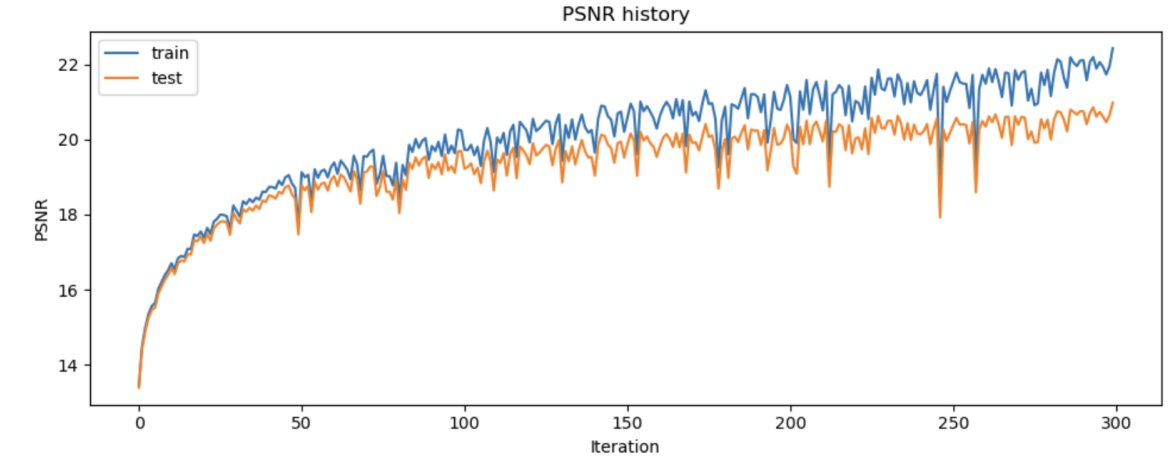
```
hidden_size = 256
epochs = 300
learning_rate = 0.2
```

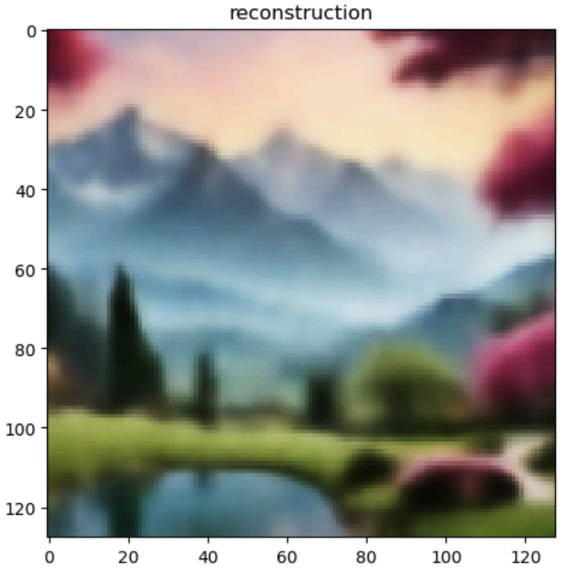


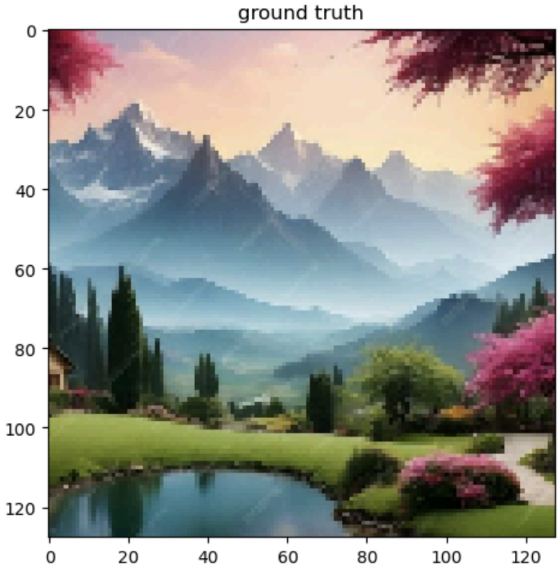
Part 3: High resolution (image of your choice)

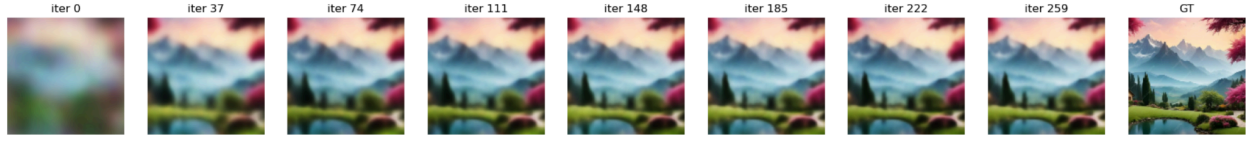
```
Hidden_size = 256
Opt = 'SGD'
Mapping = gauss_1.0
Learning_rate = 2.0
Epoch = 300
```


MSE history
MSE Loss
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0
50
100
150
200
250
300
Iteration


PSNR history
train
test
PSNR
22
20
18
16
14
0
50
100
150
200
250
300
Iteration


reconstruction
0
20
40
60
80
100
120


ground truth
0
20
40
60
80
100
120


iter 0
iter 37
iter 74
iter 111
iter 148
iter 185
iter 222
iter 259
GT

Part 4: Discussion

1. **Interesting implementation choices:** I used **mini-batch SGD (batch size = 32) with no weight decay** because it provides a balance between stable updates and faster convergence. The small batch size ensures frequent updates while maintaining some stochasticity, helping the model fit the data well. Since the goal of this project is to **overfit**, I skip weight decay to let the model fully capture patterns in the training set without any regularization constraints.

2. **Hyperparameter tuning (best parameters in bold)**

Low-resolution:

Hidden_size = **256, 512**
Epochs = [150, 300, 500, 1000, **2000,** 3000]
Learning_rate = [1e-4, 1e-5, 0.0075, **0.02**, 0.1, 0.2, 1, 2]

High-resolution:

Hidden_size = **256, 512**
Epochs = [150, **300**, 1000]
Learning_rate = [0.001, 1e-4, 0.0075, 0.004, 0.009, 0.01, 0.1, **0.2,** 0.5, 1, 2]

3. **Observations:**
    a. The model was able to converge in 150 epochs.
    b. Increasing the hidden size to 512 did not make much of a difference.
    c. The larger the learning rate, the more unstable the MSE was.
    d. The default learning rate value (=0.1) led to a straight MSE.
    e. The same set of hyperparameters worked well for both low and high-resolution image reconstruction.

How did the different choices for coordinate mapping functions compare?

- **Gauss_1.0 Mapping:** Its random nature helps capture a wider range of variations, enabling the model to recover intricate details of the ground truth image more effectively.
- **Basic Mapping:** While it uses a fixed transformation that preserves essential features, it lacks the stochastic variability of Gauss_1.0, leading to moderately lower reconstruction quality.
- **No Mapping:** This approach relies solely on the network's inherent capability, restoring the overall structure but missing finer details and nuances captured by the mapping techniques.

Do you make any interesting observations from the train and test plots?

1. Low resolution:
    - The "gauss_1.0" mapping's PSNR grows faster at the beginning, indicating it learns more rapidly in early iterations, while "none" mapping has a comparatively shallow slope, suggesting slower improvement.

2. High resolution:
    - Both train and test plots were similar in their shapes and patterns. We see that PSNR is not stable overall.
    - Performance : Gauss > Basic > None

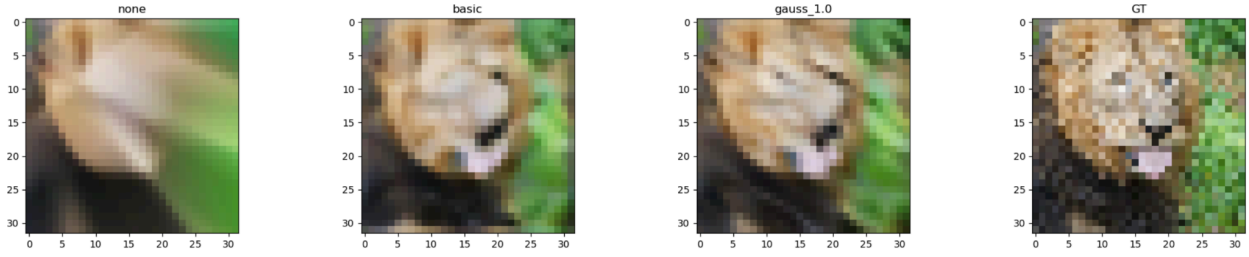
What insights did you gain from your own image example (Part 3)?

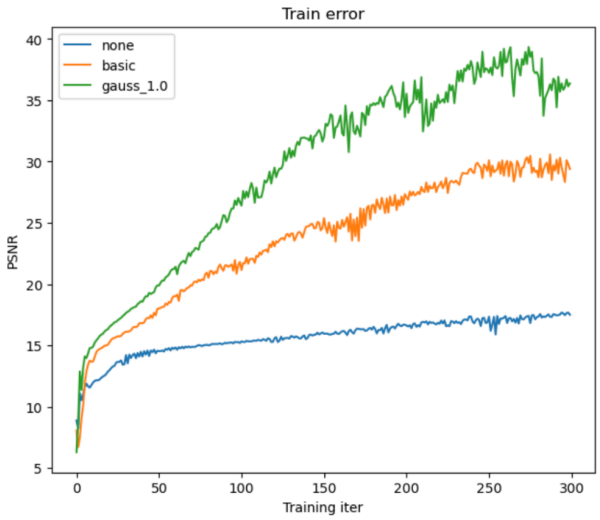
- Surprised it worked pretty well because the picture had many different objects, different shapes, and different colors. It worked well; however, some details were still missing, and the image overall was still a bit blurry.
- Surprised the gauss_1.0 was able to get the picture restored on iter 37 already.
- PSNR was a bit unstable. (Since the learning rate is chosen large)
- The reconstructed output of nature looks better than the reconstructed output of the lion, mainly because losing some details in the image of nature would sill be fine from the angle of overall picture quality but for the images that involve face, it would be difficult.

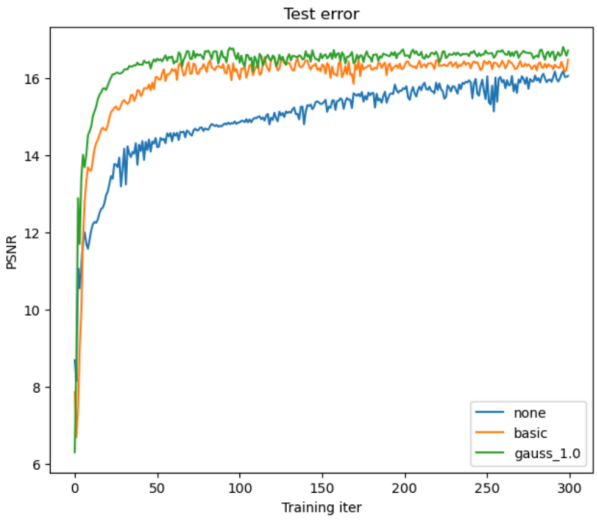
Part 5: Additional Implementations

1. Adam Optimizer

Low-resolution
Hidden_size = 256
Epochs = 300
Learning_rate = 0.0075





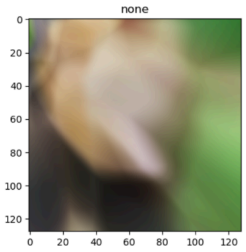


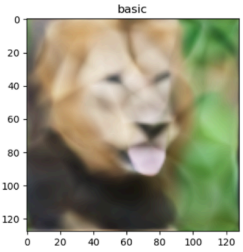
High-resolution

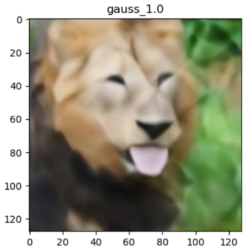
Hidden_size = 256

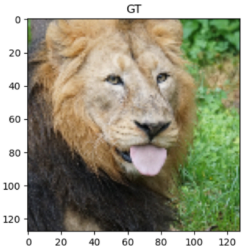
Epochs = 300

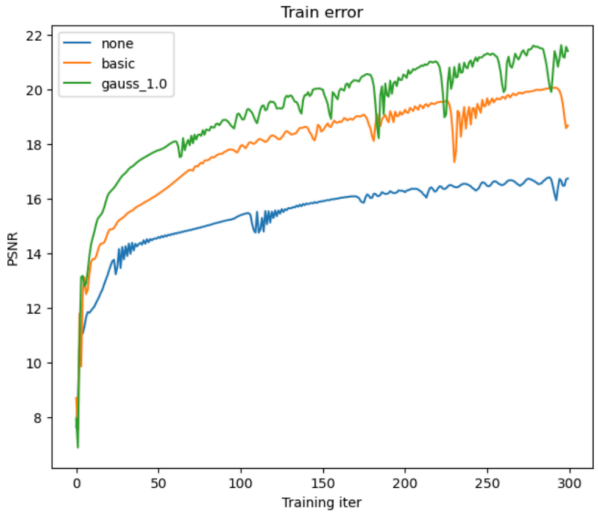
Learning_rate = 0.0075

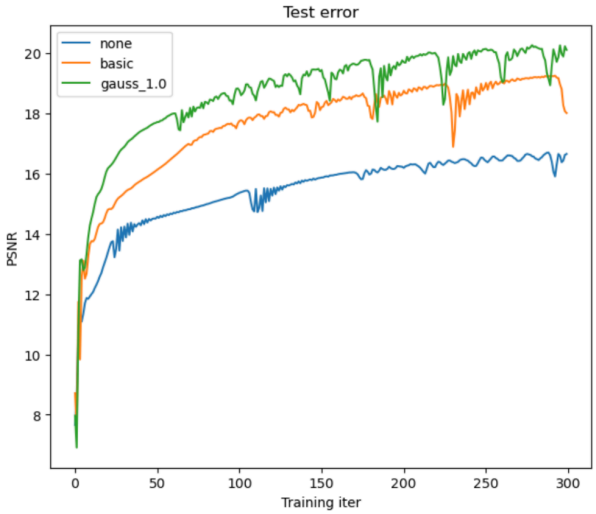








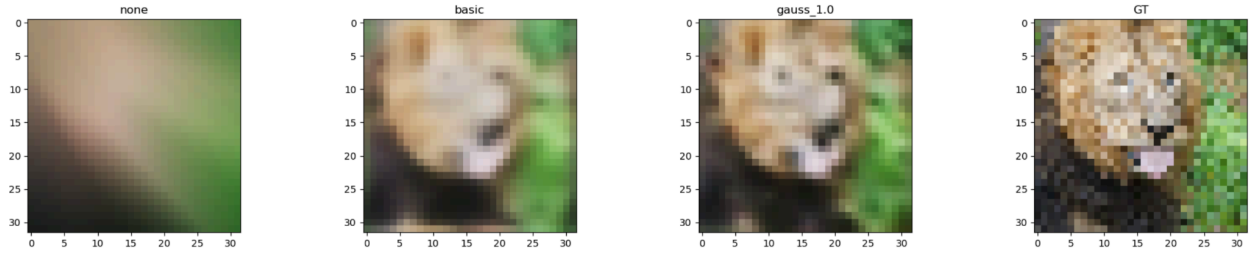


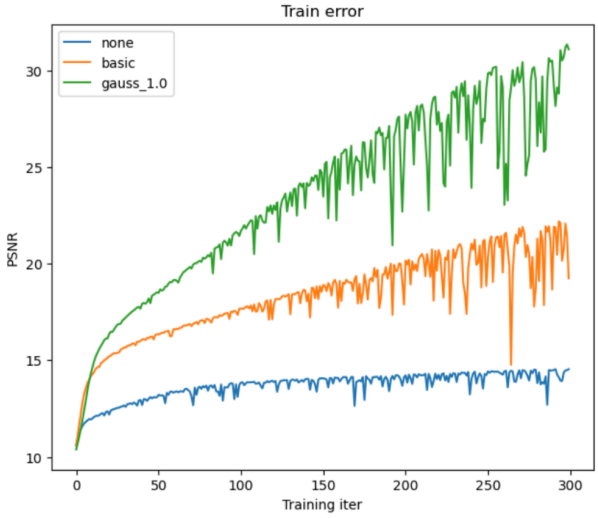


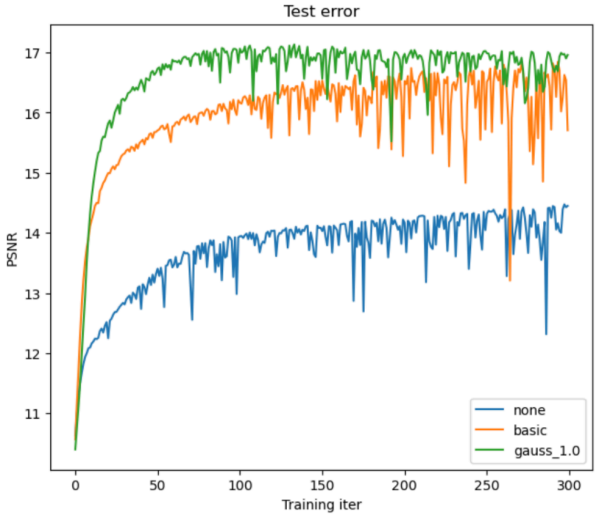
2. Deeper Network

Low resolution

Hidden_size = **768**
Epochs = 300
Learning_rate = 0.2
Opt = 'SGD'
Number of layers = 6

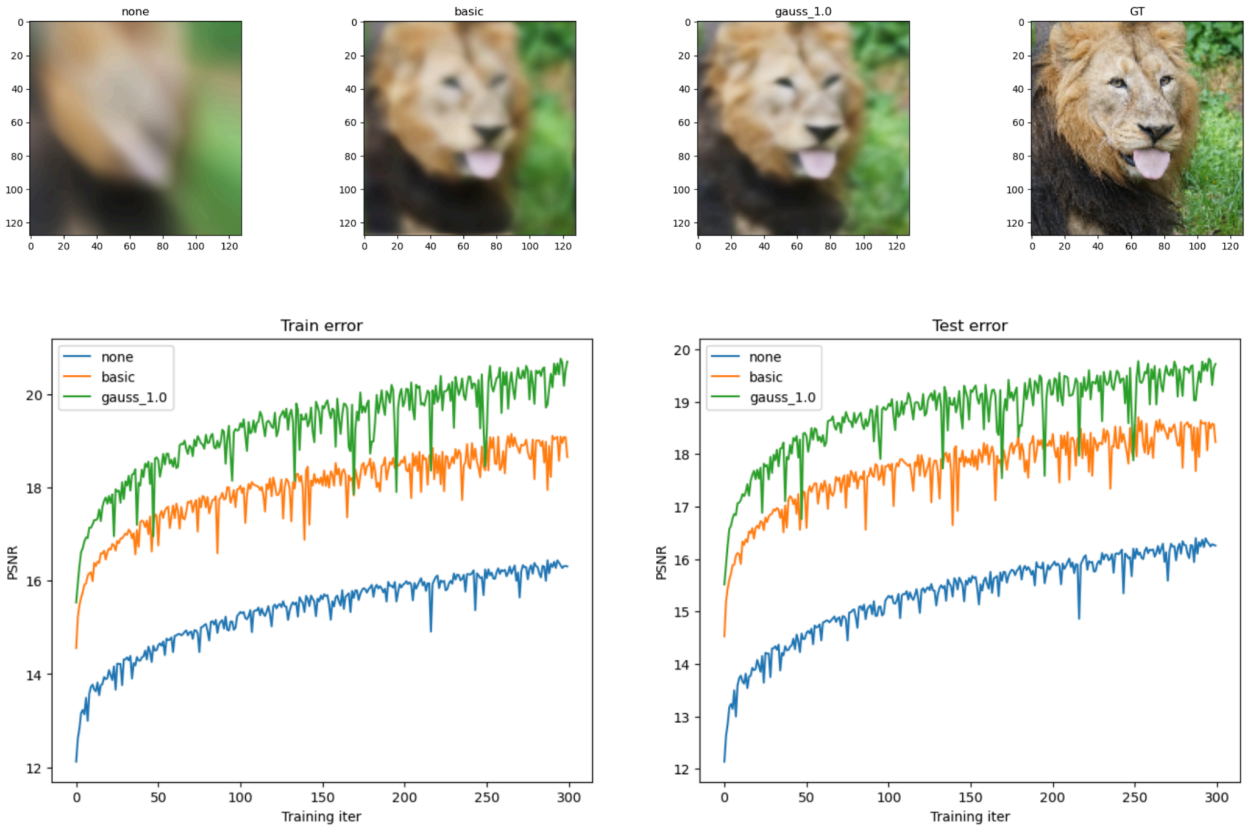






High resolution

```
Hidden_size = 768
Epochs = 300
Learning_rate = 0.2
Opt = 'SGD'
Num_layers = 6
```

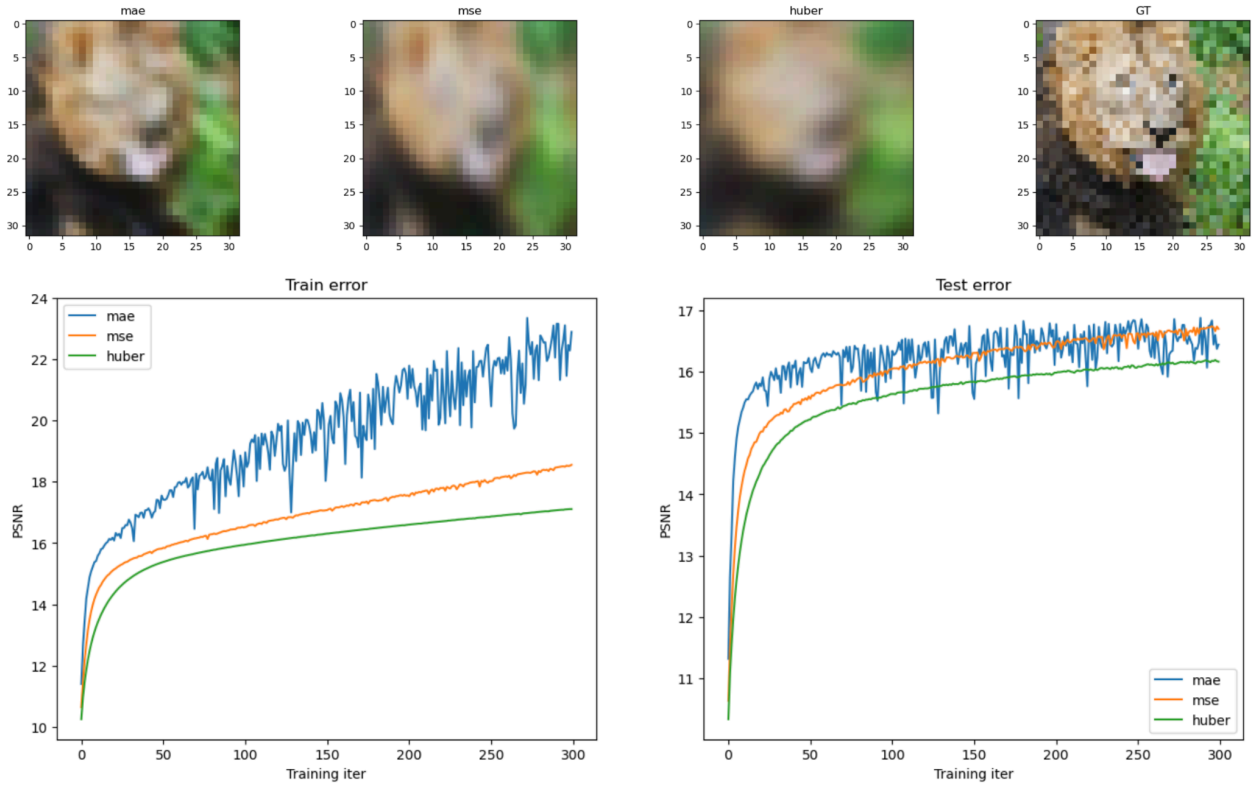


3. Alternative Losses

Mean Absolute Error (L1 Loss): Penalizes all errors equally, encouraging sharp, less blurry reconstructions. It's less affected by outliers than MSE, resulting in clearer edges but sometimes noisier images.

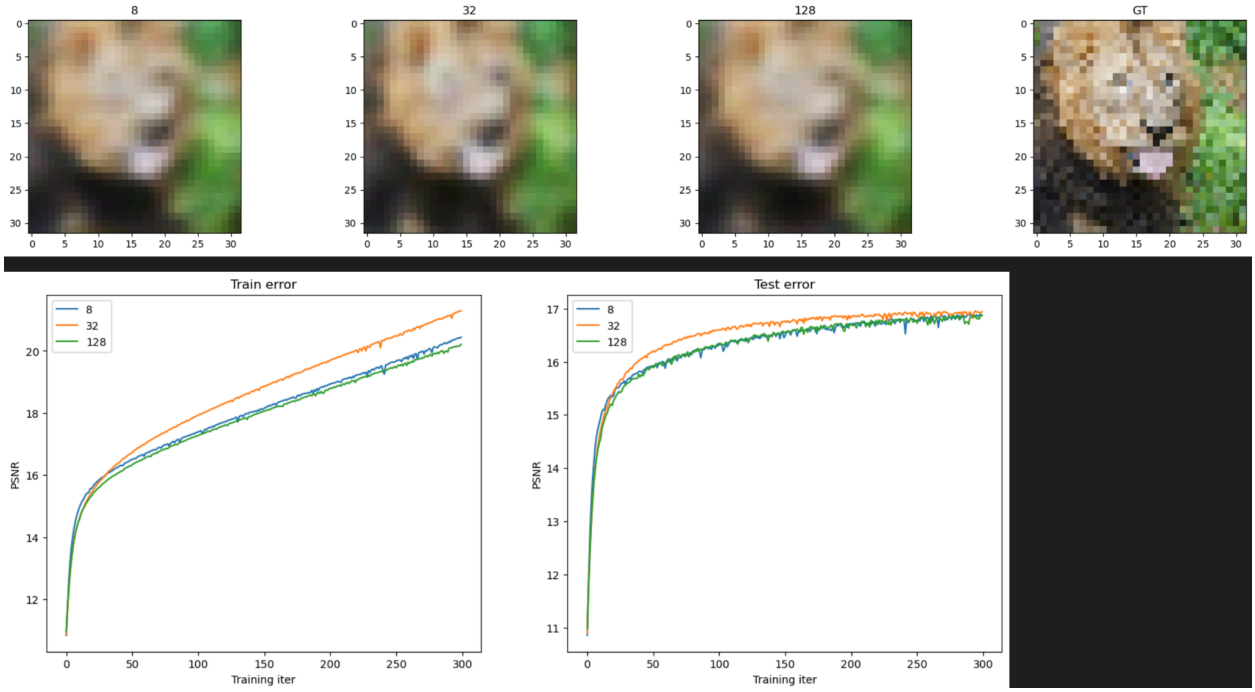
Huber Loss: Blends MSE and MAE by acting quadratic for small errors (smooth optimization) and linear for large errors (robustness), making it ideal for balancing sharpness and stability.

From the graph, we see that the PSNR value of Huber is smaller than that of MSE and MAE. Also, the MAE is more unstable than the MSE. From the following qualitative examples, it's evident that the MAE gives a sharper image than the MSE, and Huber gives the most blurred image.



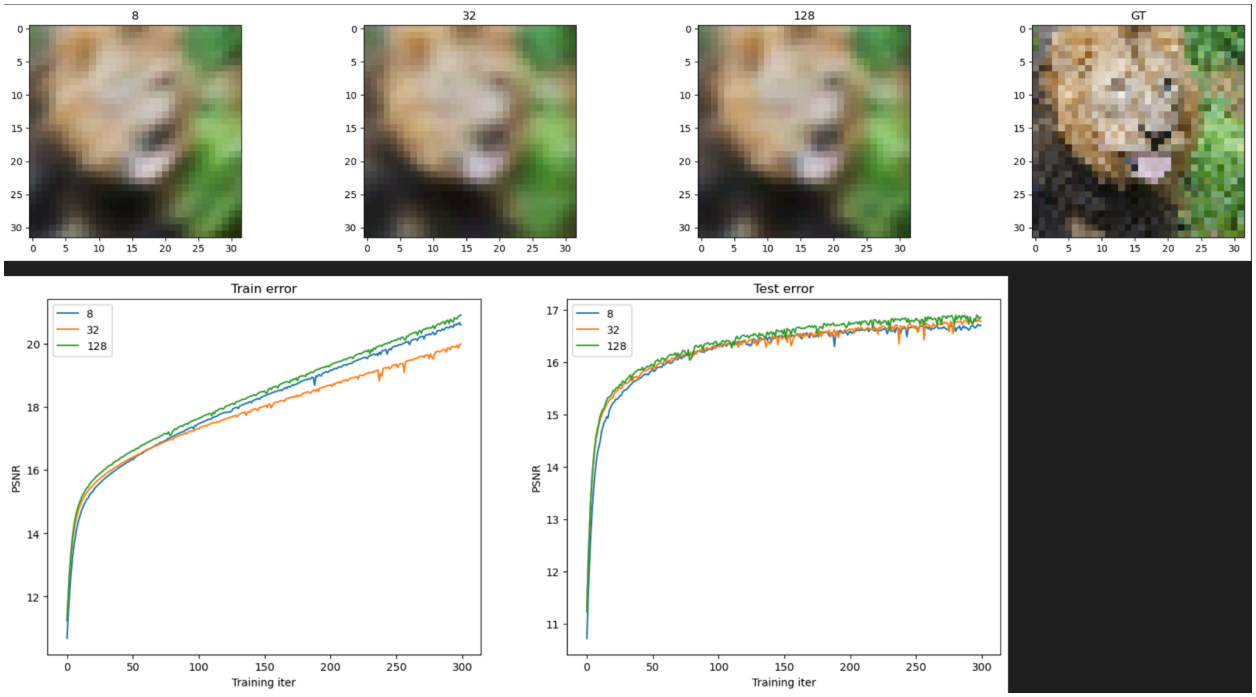
4. Gaussian Fourier Feature mapping hyperparameters

Mapping size = [8, 32, 128]; Gaussian Scale = [1]



Here, the mapping size increases from left to right.

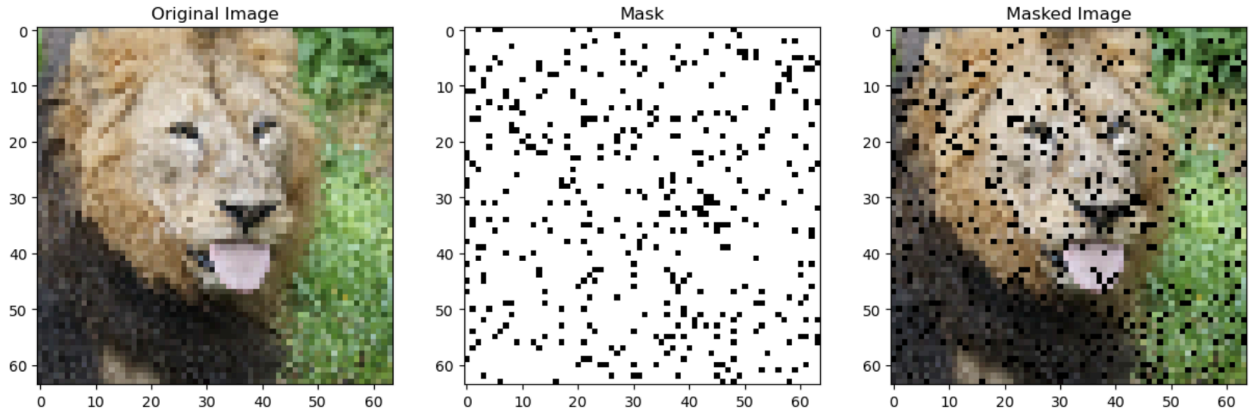
Mapping size = [8, 32, 128]; Gaussian Scale = [2]

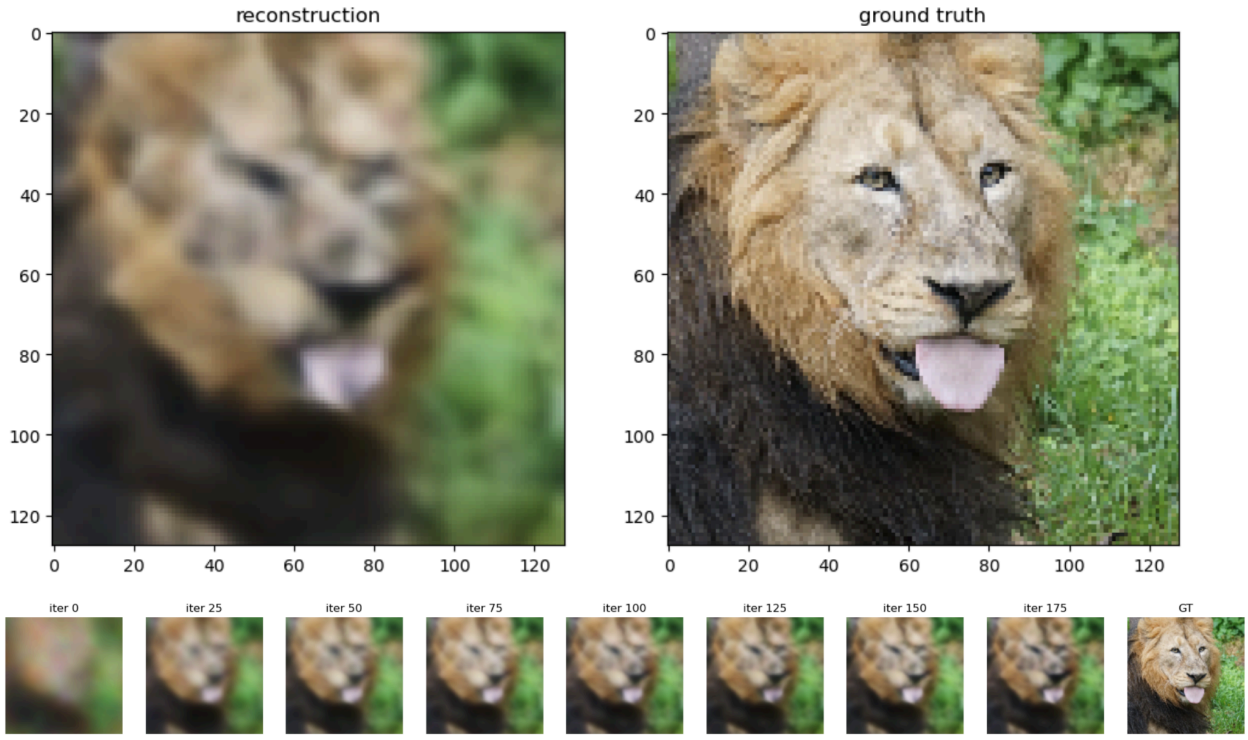


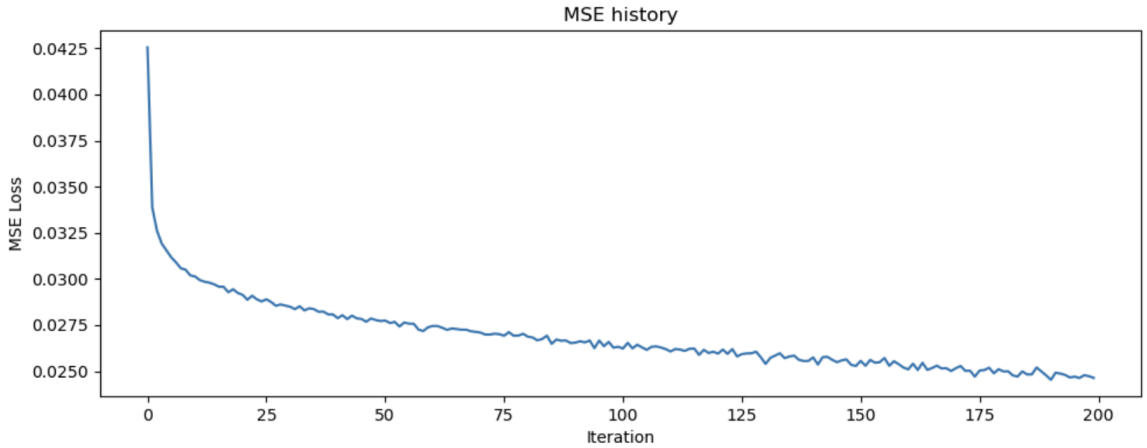
As can be observed through the above two figures, as the mapping size increases, the reconstructed images become sharper and better capture high-frequency details, resulting in reduced blur. Conversely, smaller mapping sizes usually produce blurrier reconstructions due to limited frequency representation.

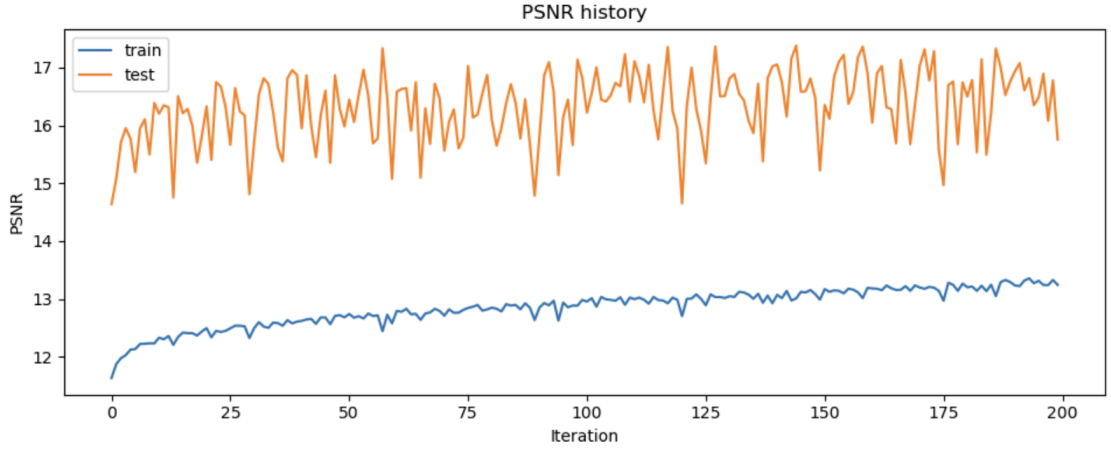
5. Interesting image inpainting or restoration scenarios

```
Hidden_size = 256
Epochs = 300
Learning_rate = 0.2
Mapping = 'gauss_1.0'
Opt = 'SGD'
```




reconstruction
ground truth
iter 0
iter 25
iter 50
iter 75
iter 100
iter 125
iter 150
iter 175
GT


MSE history
MSE Loss
Iteration


PSNR history
train
test
PSNR
Iteration