MIC A1
Q1-3

Gautam Siddharth K - 23B0957
Evuri Mohana Sreedhara Reddy - 23B1017

Rician Likelihood Model

Objective Function

$$\max_{x_i} P(x|y,\theta) = \min_{x_i} \left(\frac{y_i^2 + x_i^2}{2\sigma^2} - \log I_0\left(\frac{y_i x_i}{\sigma^2}\right) + \sum_{a \in A_i} V_a(x_a) \right)$$

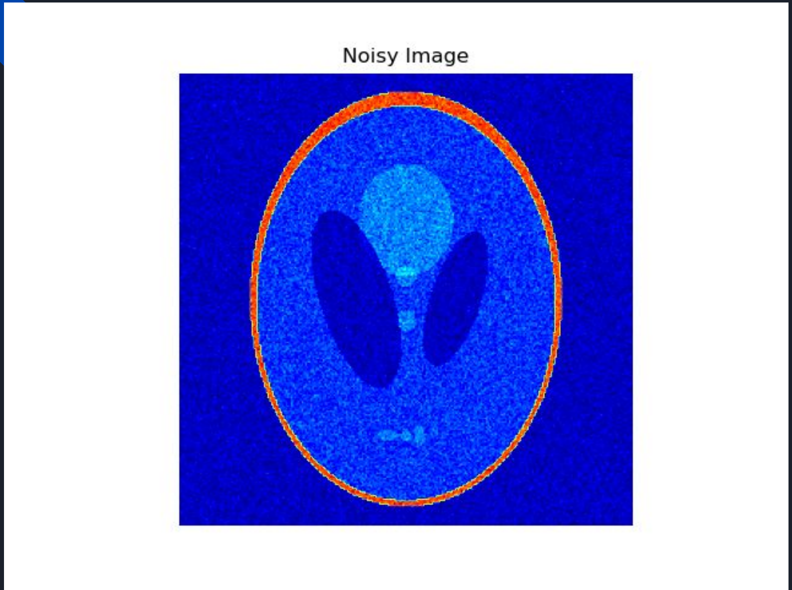
OR

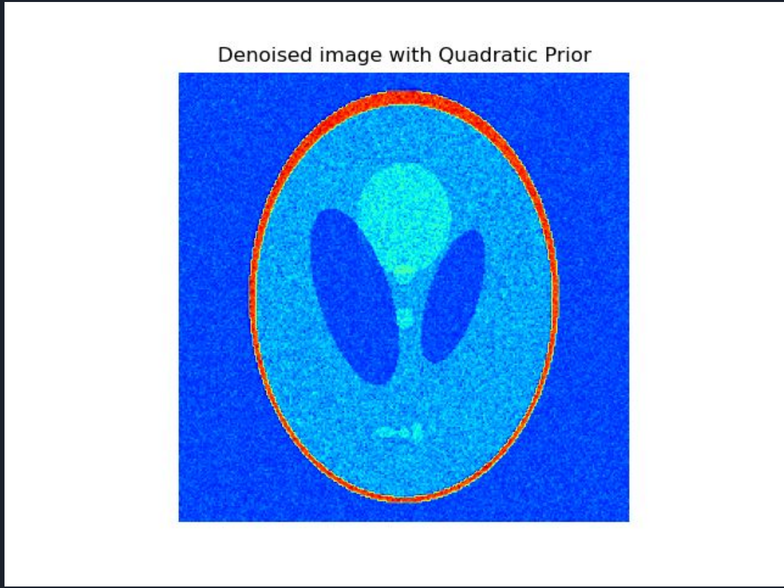
Gradient

-log

$$\frac{\partial P(x|y,\theta)}{\partial x_i} = \frac{x_i}{\sigma^2} - \frac{I_1\left(\frac{y_i x_i}{\sigma^2}\right)}{I_0\left(\frac{y_i x_i}{\sigma^2}\right)} \frac{y_i}{\sigma^2} + \frac{\partial}{\partial x_i} \sum_{a \in A_i} V_a(x_a)$$

Q1A Quadratic Prior



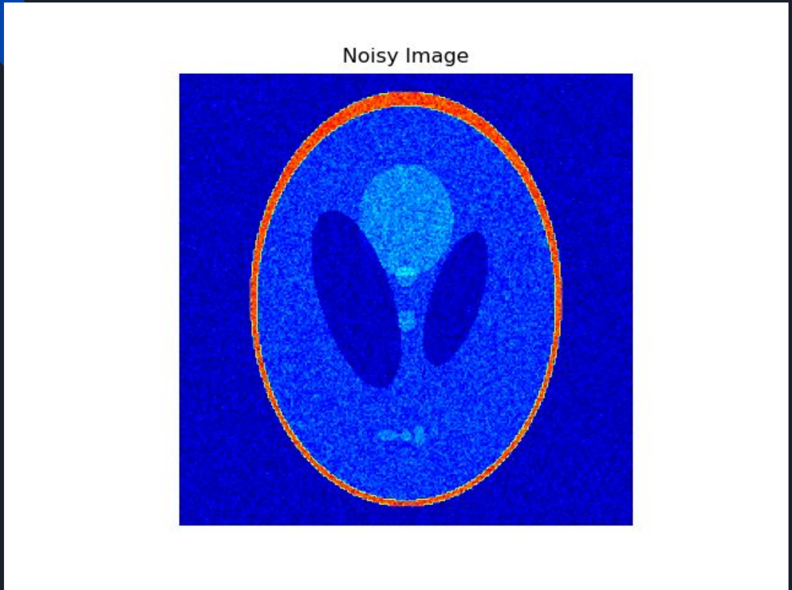


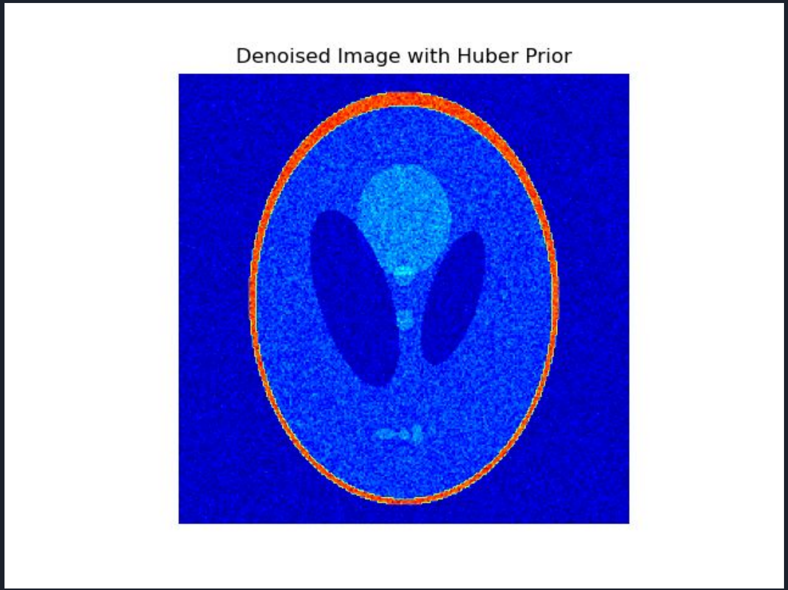
<u>Parameters</u>
Alpha = 0.5
Initial Learning Rate = 0.01
LR Decay = 0.05

Noisy RRMSE: 0.298
Denoised RRMSE: 0.277

+20% RRMSE: 0.281
-20% RRMSE: 0.280

Q1B Huber Prior





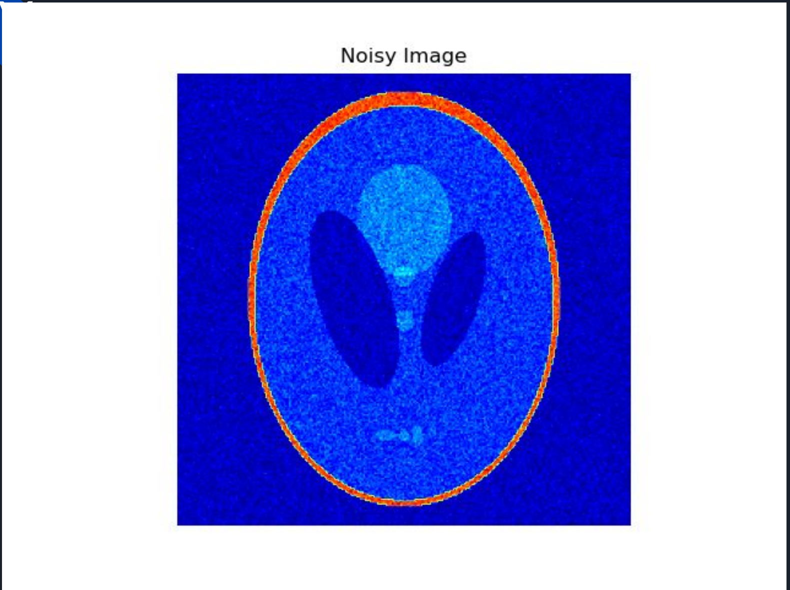
Parameters
Alpha = 0.99
Initial Learning Rate = 0.01
LR Decay = 0.05

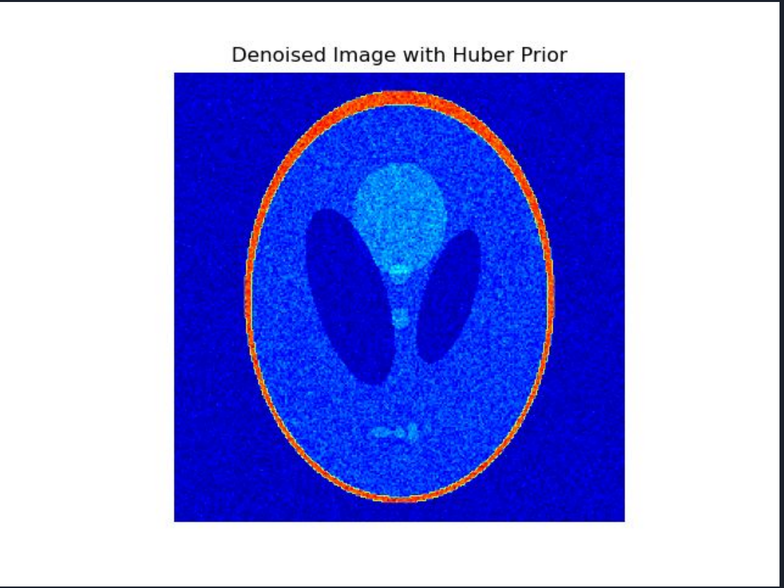
Noisy RRMSE: 0.298
Denoised RRMSE: 0.264

+20% alpha RRMSE: 0.267
-20% alpha RRMSE: 0.266

+20% gamma RRMSE: 0.284
-20% gamma RRMSE: 0.268

Q1C Discontinuous Adaptive Prior



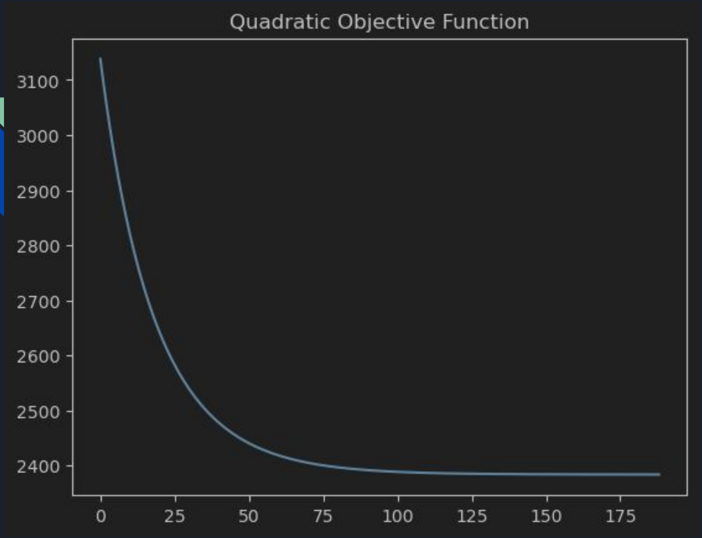


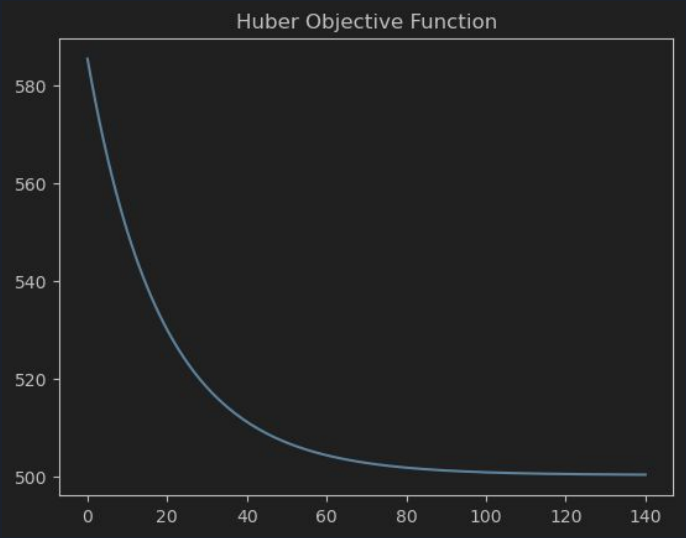
Parameters
Alpha = 0.99
Initial Learning Rate = 0.01
LR Decay = 0.05

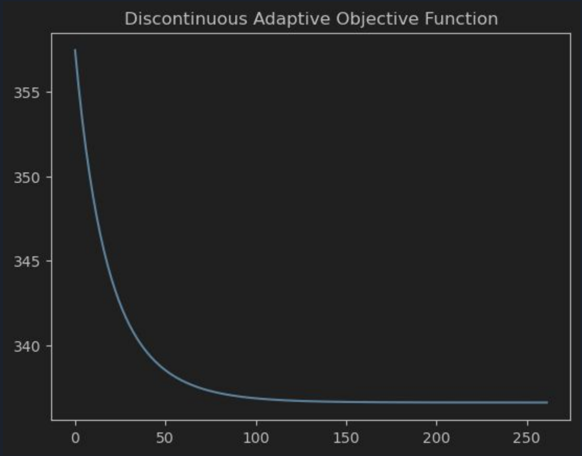
Noisy RRMSE: 0.298
Denoised RRMSE: 0.235

+20% alpha RRMSE: 0.235
-20% alpha RRMSE: 0.287

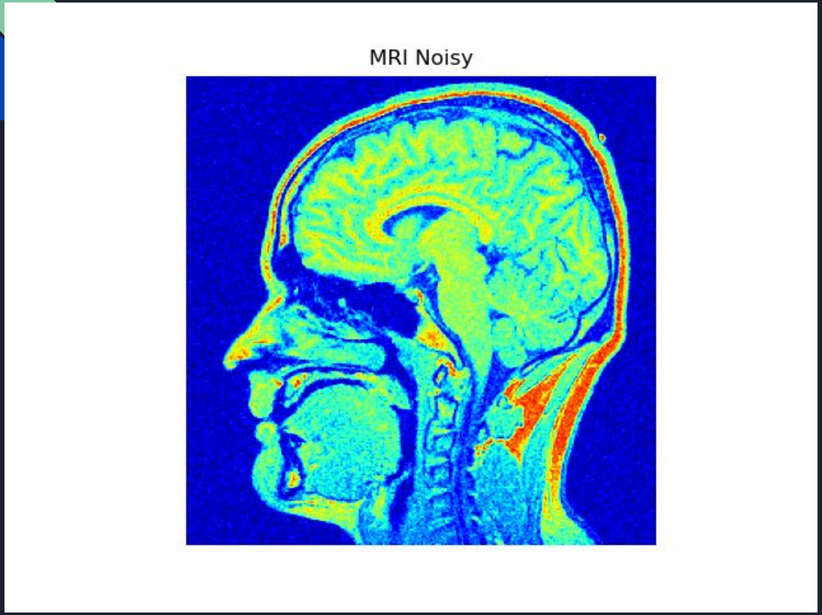
+20% gamma RRMSE: 0.284
-20% gamma RRMSE: 0.268

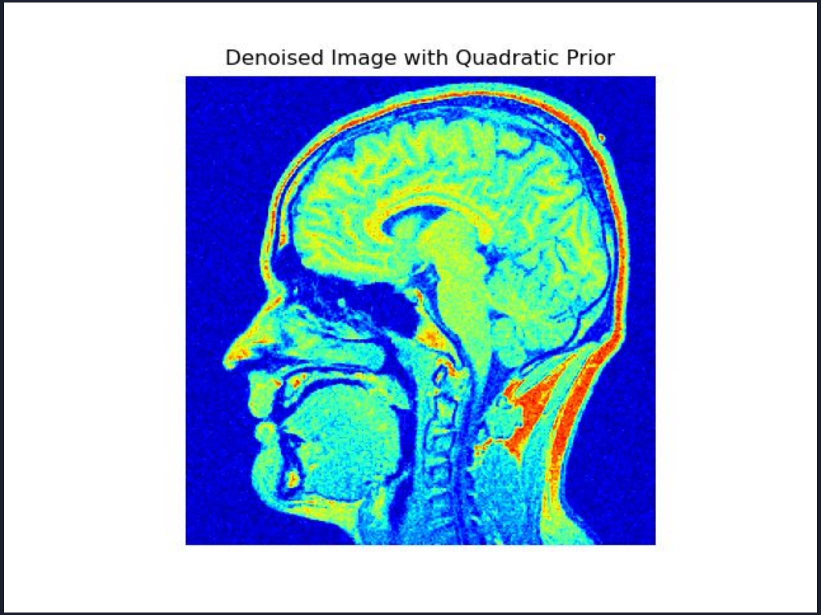

Quadratic Objective Function
3100
3000
2900
2800
2700
2600
2500
2400
0
25
50
75
100
125
150
175


Huber Objective Function
580
560
540
520
500
0
20
40
60
80
100
120
140


Discontinuous Adaptive Objective Function
355
350
345
340
0
50
100
150
200
250

Q2A) Quadratic Prior



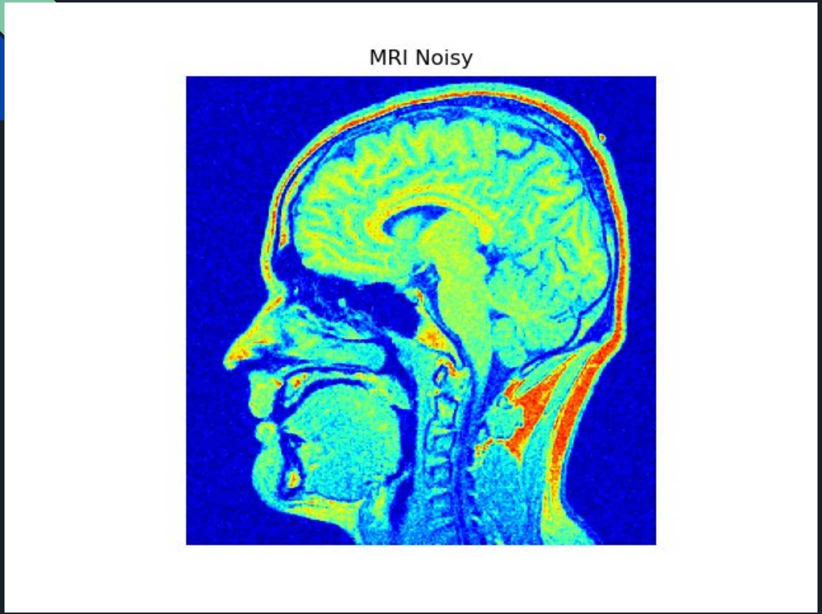


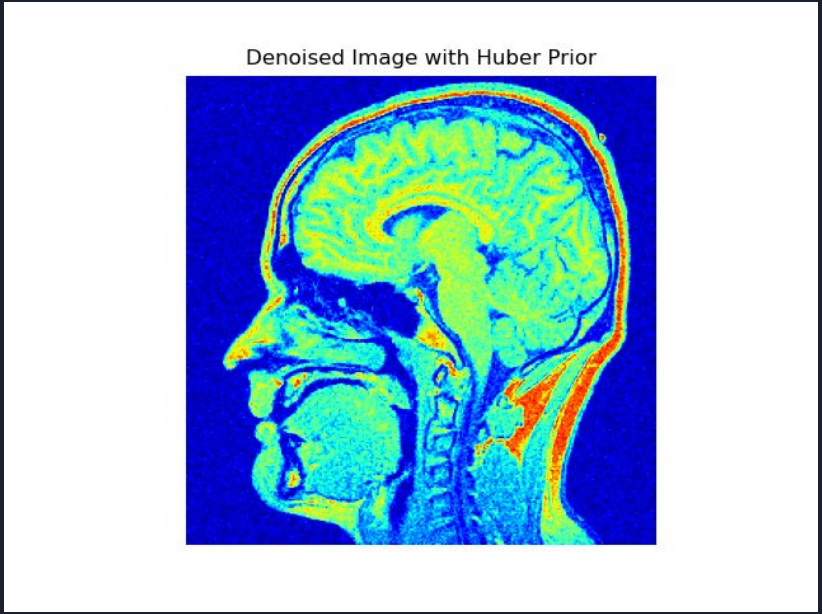
<u>Parameters</u>
Alpha = 0.8
Initial Learning Rate = 0.005
LR Decay = 0.05

Noisy RRMSE: 0.142
Denoised RRMSE: 0.149

+20% alpha RRMSE: 0.155
-20% alpha RRMSE: 0.144

Q2B) Huber



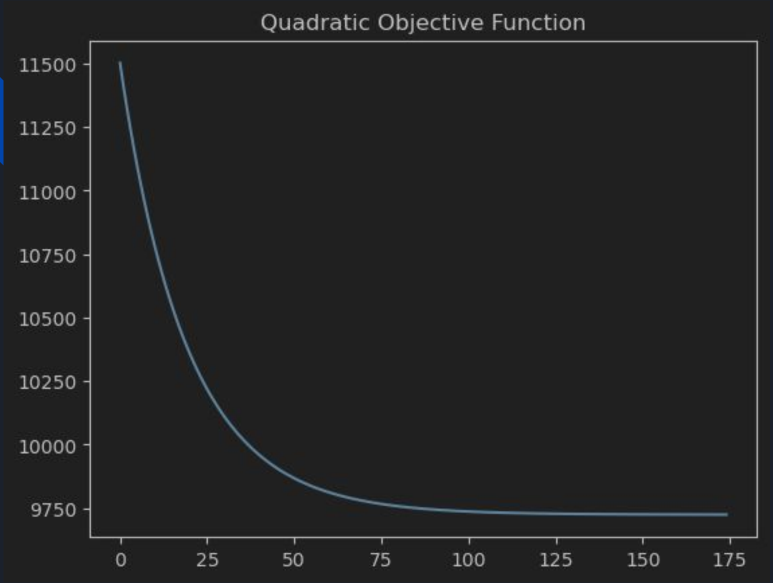


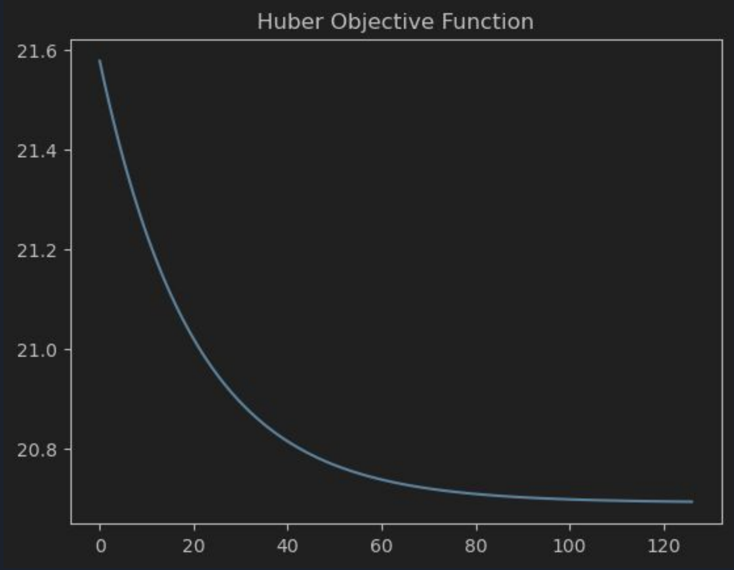
Parameters
Alpha = 0.8
Initial Learning Rate = 0.005
LR Decay = 0.05

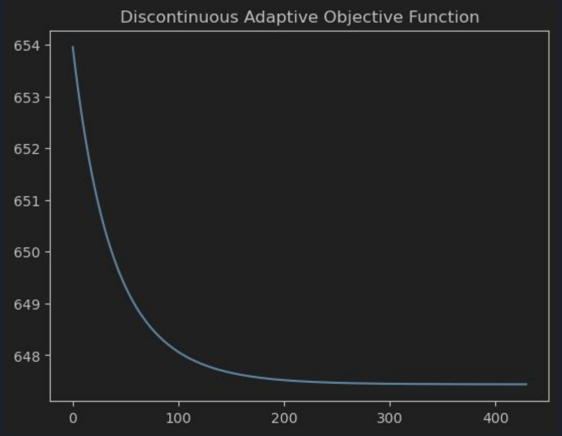
Noisy RRMSE: 0.142
Denoised RRMSE: 0.137

+20% alpha RRMSE: 0.138
-20% alpha RRMSE: 0.139

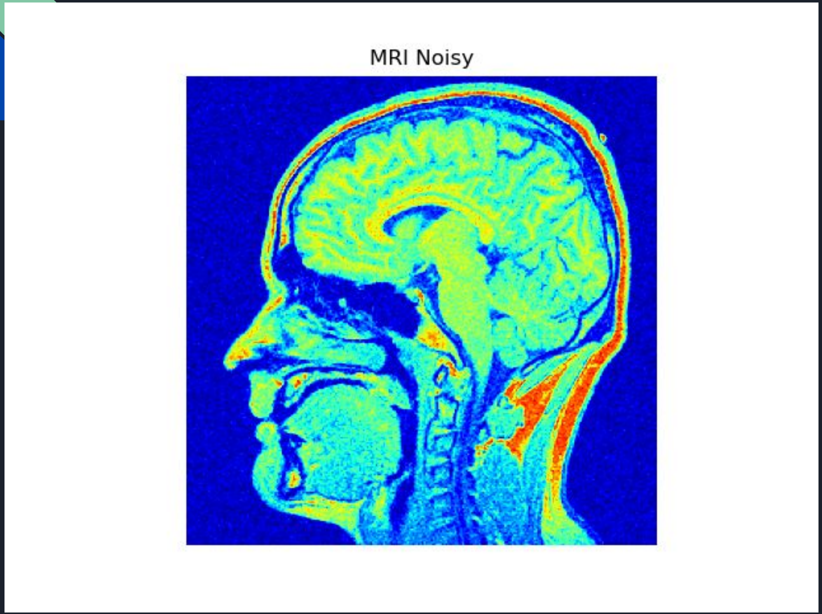
+20% gamma RRMSE: 0.142
-20% gamma RRMSE: 0.139

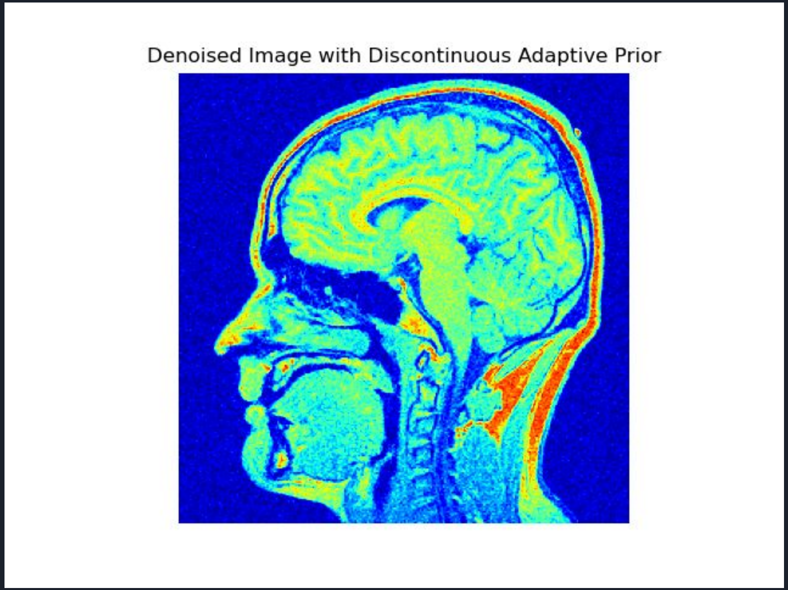

Quadratic Objective Function
11500
11250
11000
10750
10500
10250
10000
9750
0
25
50
75
100
125
150
175


Huber Objective Function
21.6
21.4
21.2
21.0
20.8
0
20
40
60
80
100
120


Discontinuous Adaptive Objective Function
654
653
652
651
650
649
648
0
100
200
300
400

Q2C) Discontinuous Adaptive



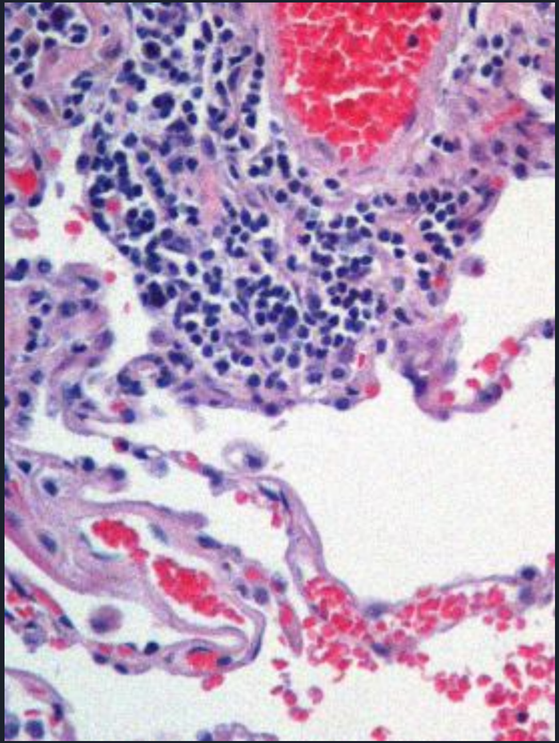


Parameters
Alpha = 0.95
Initial Learning Rate = 0.005
LR Decay = 0.02

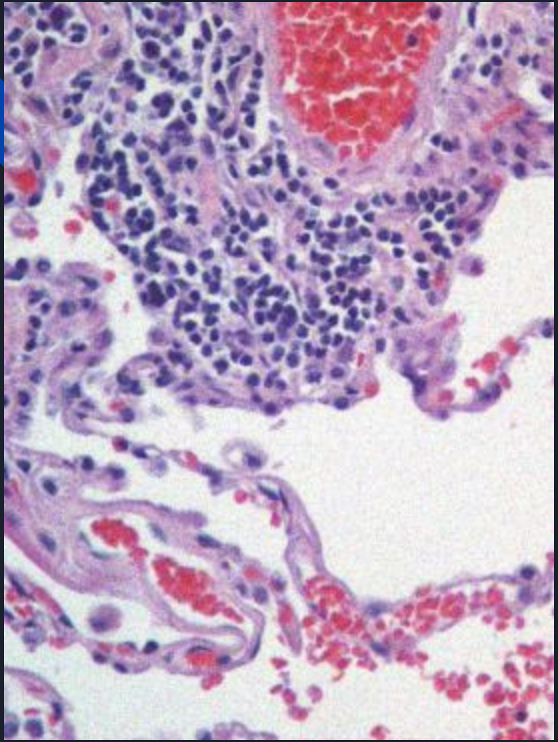
Noisy RRMSE: 0.142
Denoised RRMSE: 0.133

+20% alpha RRMSE: 0.133
-20% alpha RRMSE: 0.148

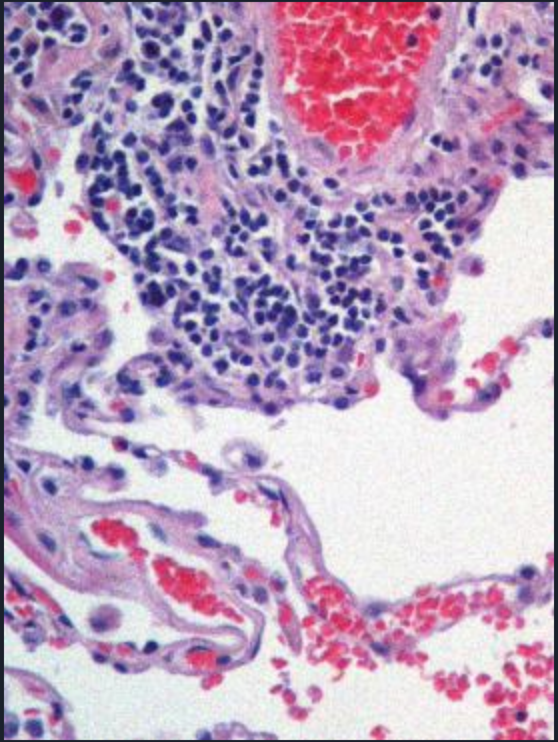
+20% gamma RRMSE: 0.145
-20% gamma RRMSE: 0.166



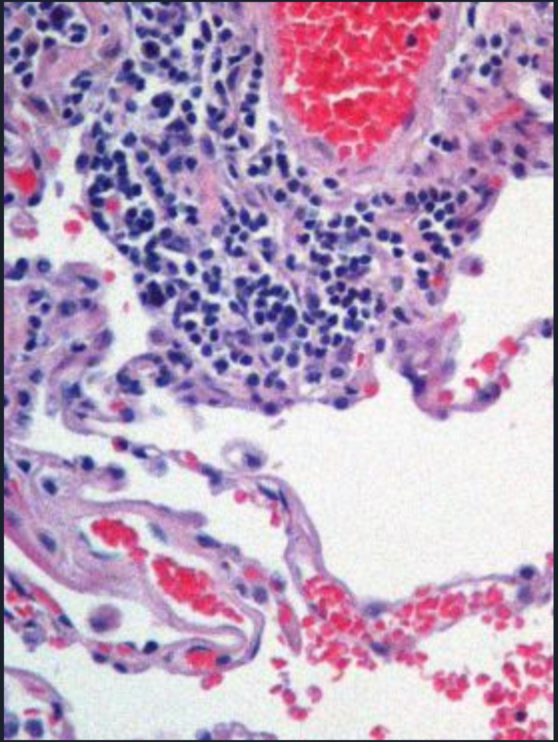
Noisy
RRMSE: 3.69



Sq l2 norm
RRMSE 3.78



L2 norm
RRMSE: 3.694



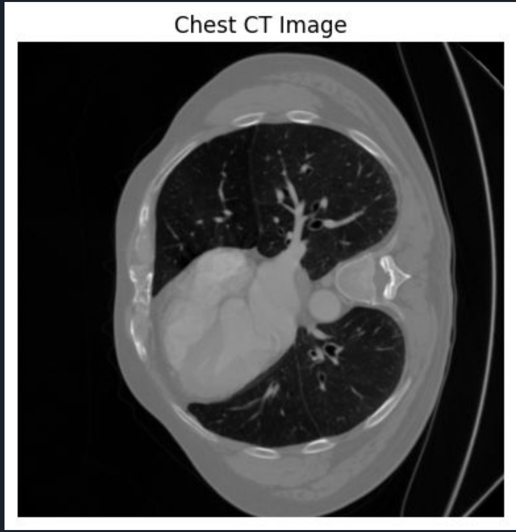
Huber norm
RRMSE: 3.694

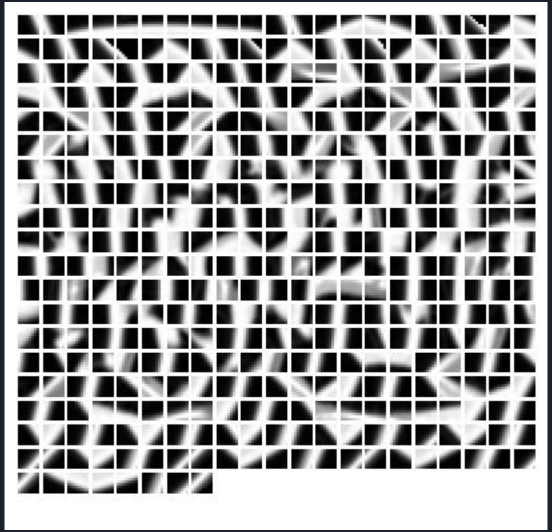
MIC Assignment 1 Q4

Evuri Mohana Sreedhara Reddy - 23B1017
Gautam Siddharth K - 23B0957

The Patches extracted from the image

- We are extracting the 8x8 high variance patches from the image we have and are storing them in an array (X).
- The threshold variance used = 0.02.





The Dictionary Learning

- We would like to minimize the objective function: $\arg\min_{D} \min_{r} \sum_{i=1}^{I} \|x_i - Dr_i\|_2^2 + \lambda \|r_i\|_p^p$

- For this, we are going to use gradient descent. We initialize the matrix **D** with SVD, but it really doesn't matter, we can start with a random matrix.
- Now, in every iteration, we try to find the best values of **r** for which the coefficients are a good fit for the matrix **X**, in the function `sparse_coding`.
- For the achieved value of **r**, we update the matrix **D** using the `update_dictionary` function.
- Now, this process is looped. (for 500 iterations in our case)

The gradient functions - Math

- `update_dictionary`: The update for **D**, with a given **r** is the learning rate times the gradient. We normalize at each of the steps: the gradient and the new dictionary.

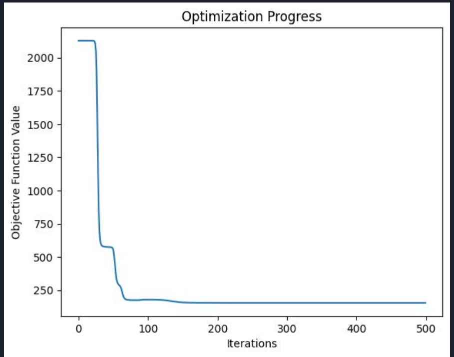
$$\nabla_D = \sum_i (x_i - Dr_i)r_i^T$$

- `sparse_coding`: For the value of r, given D, we start off with a coeff matrix of zeros, run a nested learning model for r, using gradient descent. (with 50 iterations in our case)

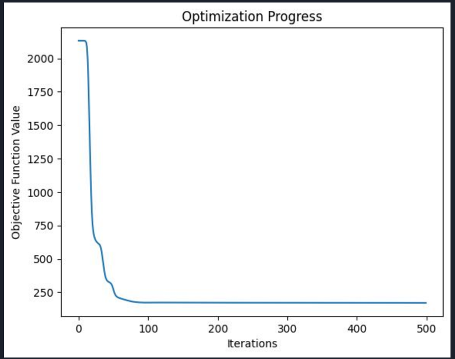
$$\nabla_r = -2D^T(X - Dr) + p\lambda \cdot (\mathrm{sign}(r) \odot |r|^{p-1})$$

The learning curves for different norms

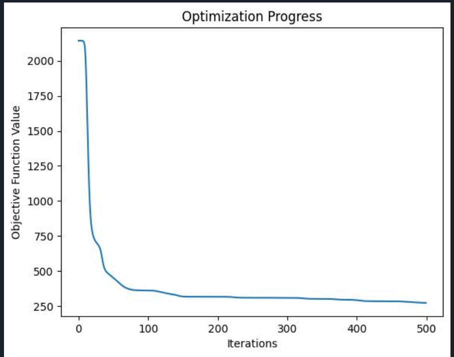
p = 2



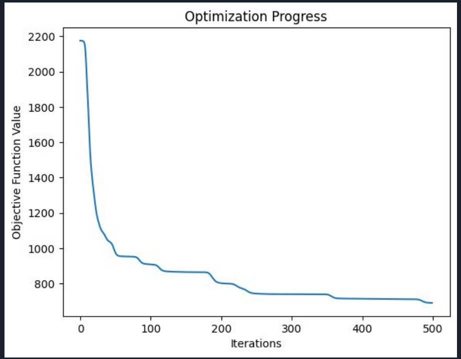
p = 1.6



p = 1.2



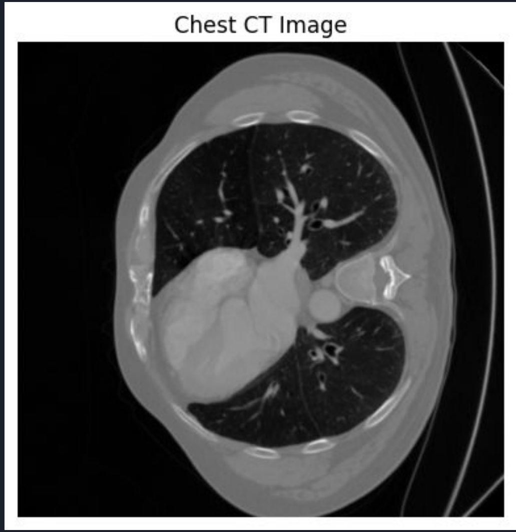
p = 0.8

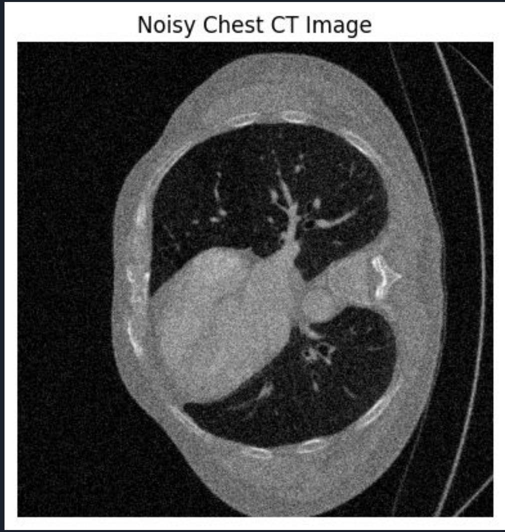


- From the graphs, it seems like the learning is slower for the smaller values of p, compared to the larger ones.

Noising an Image

- Used the basic Gaussian noising, 10% of the range, and applied the bounds of the values to be in between 0 and 1.





Denoising by learning from the patches

- We use the dictionary learnt for p = 0.8 to denoise the noisy image generated by the gaussian.
- The crude idea is as follows:
  - We divide the image into 8x8 patches, with an overlap, (we can take 50%). Now, we try to learn r this time from the previous learning algorithm, instead of the D. We can do this by sparse coding.
  - Now, the new patches can be represented as D*r. We revert the denoised image by putting back the patches by multiplying the weights by the fraction (take care of the boundary cases).
  - Finally, limit the values of the image to between 0 and 1.