# Assignment 1

## Introduction:

We are provided with two datasets of CT images. One is low dose (LDCT) and the other is called normal dose (NDCT). Like the name suggests, these datasets are captured with different dose. Dose usage directly impacts the image quality i.e., higher the dosage lesser the Noise in image. Our task is to apply different image processing techniques to improve image quality of low dose images and close the gap between higher dose images. Assignment is divided into three parts which are as follows

1. Apply different filters to reduce noise in the low dose images.
2. Run and understand the code of dictionary learning provided by **Scikit** python library.
3. Apply dictionary learning to improve LDCT images quality.

## Part1:

In the first part we were required to apply following filters on each image:

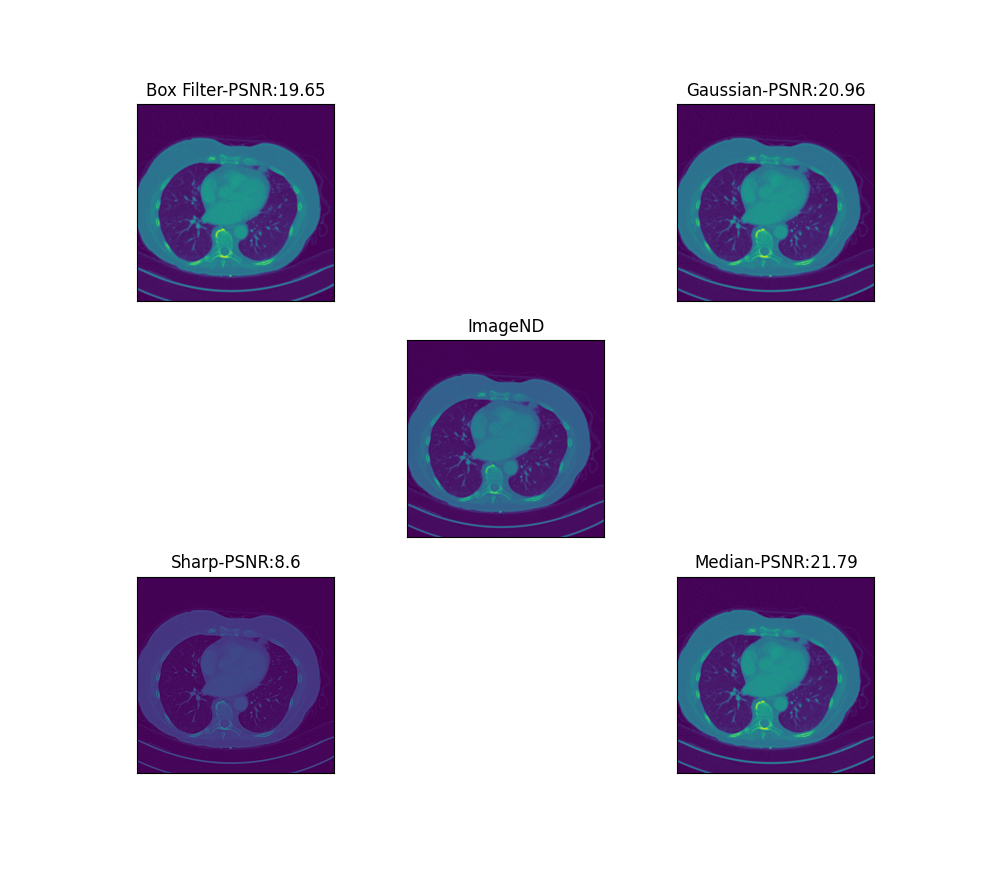
1. Box Filter
2. Gaussian Filter
3. Sharpening Filter
4. Median Filter

For comparison reasons all filters were used with a 3x3 kernel size. It was found that using larger size kernel over smoothing of images which results in loss of image quality.

### Results:

Out of all four filters, median filter performed the best and gaussian filter came second. Results were compared to NDCT images using PSNR scores. Results are provided in the CSV file containing several columns which are as follows:

* LD\_name: LDCT image name
* ND-LD\_psnr: PSNR score between ND and LD CT image as a baseline.
* Box\_psnr: PSNR score between ND and Box filter applied on LD image.
* gaus\_psnr: PSNR score between ND and Gaussian filter applied on LD image.
* sharp\_psnr: PSNR score between ND and Sharpness filter applied on LD image.
* med\_psnr: PSNR score between ND and Median filter applied on LD image.



## Part2:

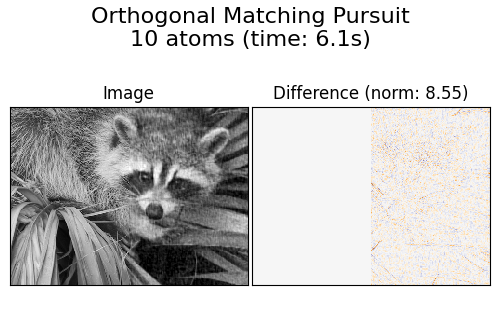
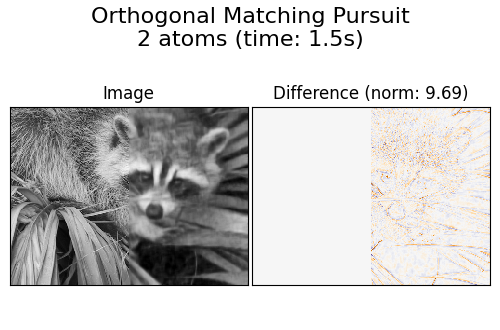
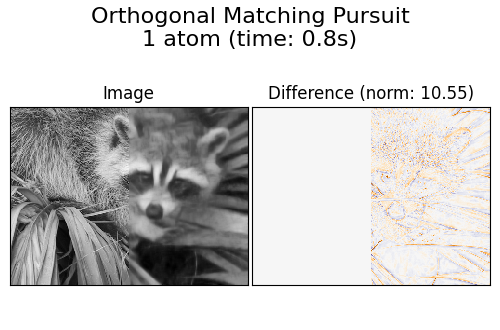
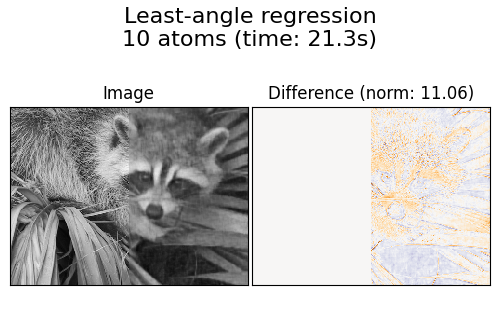
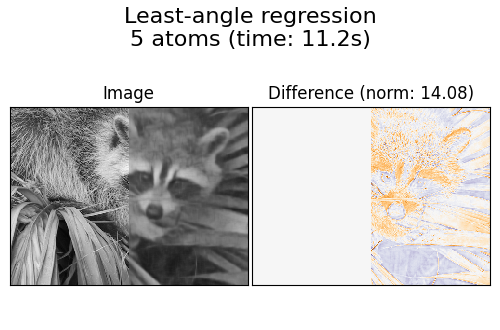
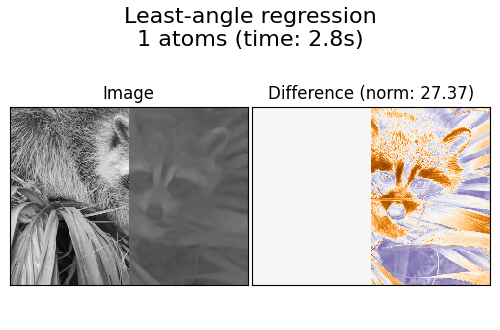
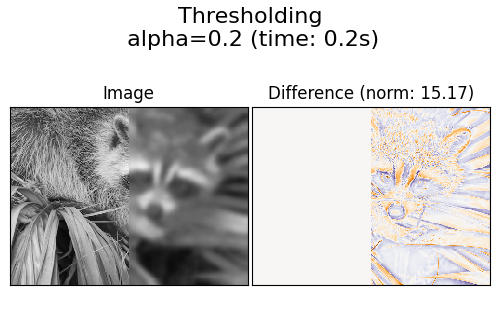
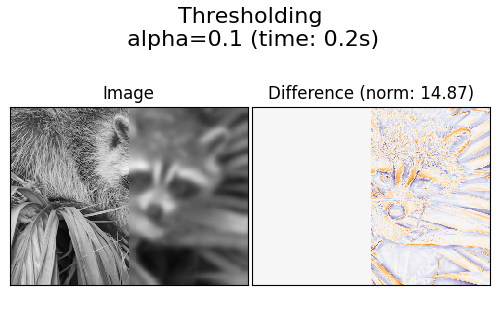
Link given in second part of assignment shows an example of dictionary learning and applying different transformation to the test data using learned dictionaries.

### Steps:

1. A racoon image is loaded, normalized and rescaled to save computation.
2. A distorted image is created by applying random noise to the right half of the original image.
3. Patches are extracted from right side of distorted image as it was left unchanged.
4. These patches are reshaped, subtracted from their norm and divided by their standard deviation.
5. Dictionaries are learnt from extracted patches.
6. Different transformation algorithms are used to estimate sparse solution.
7. These sparse codes are matched with the learnt dictionary and reconstructed patches are created.
8. Finally, image is created using reconstructed patches and compared with original image.

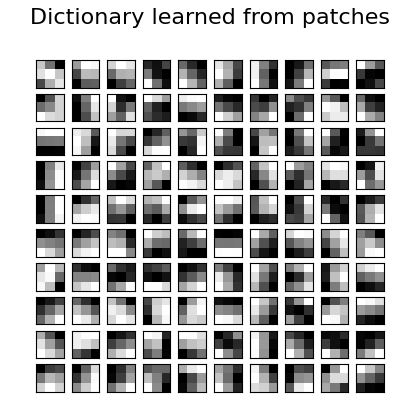
### Results:

According to the link provided, a common way to evaluate the results of image denoising is to compare the results of original and reconstructed image. For reconstruction, users have the option to use different transformation functions which are described below:

* Orthogonal Matching Pursuit (OMP): OMP starts the search by finding the column of A matrix with maximum correlation with measurement y. Here users have an option of selecting number of non-zero coefficients. This means that OMP can approximate the optimum solution vector with defined number of non-zero elements. Following figures show the influence of number of atoms on image quality. By increasing number of atoms, we get better results but there is a diminishing return.
* Least angle regression: At each step, it finds the feature most correlated with the target. When there are multiple features having equal correlation, instead of continuing along the same feature, it proceeds in a direction equiangular between the features. Changing the number of atoms improves image quality up to certain extent. Following figures summarize the results. 
* Thresholding: Thresholding is clearly not useful for denoising, but it is here to show that it can produce a suggestive output with very high speed, and thus be useful for other tasks such as object classification, where performance is not necessarily related to visualization.

## Part3:

In this part users were required to use dictionary learning for improving quality of LD images. First all the dictionaries were trained using ND images. Then these learnt dictionaries were used on LD images for image denoising. Dictionaries learnt can also be visualized below:



### Procedure:

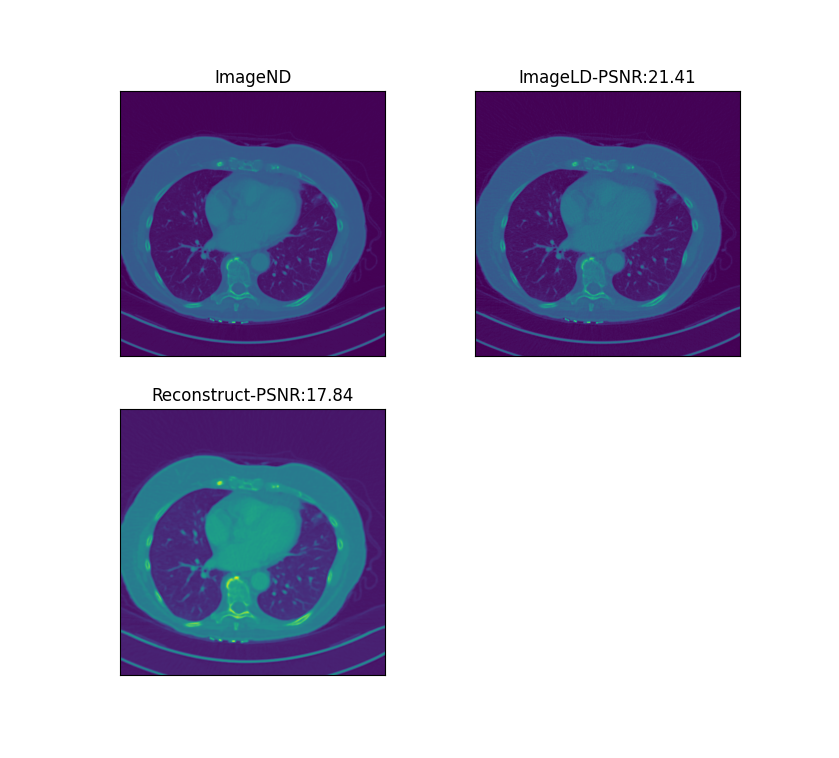
For training dictionaries, I explored two approaches which are as follows:

* **Partial\_fit:** For larger datasets, it is difficult to learn dictionaries by concatenating patches of all the dataset images as it requires a lot of memory resources. Users have the option to use partial\_fit method when training data is large. Using this method, we can extract small number of patches from each image but can iterate over all of the dataset and learn from all of the dataset.
* **Fit:** For single images, normal fit method is used and all the extracted patches from image are processed in one go. Problem with this approach is that for every new image it learns a new dictionary and we cannot use previously learnt dictionary as a starting point for the next image.

### Result:

###### *Default Parameters(Patch Size=(7x7), Num of Dictionaries=100):*

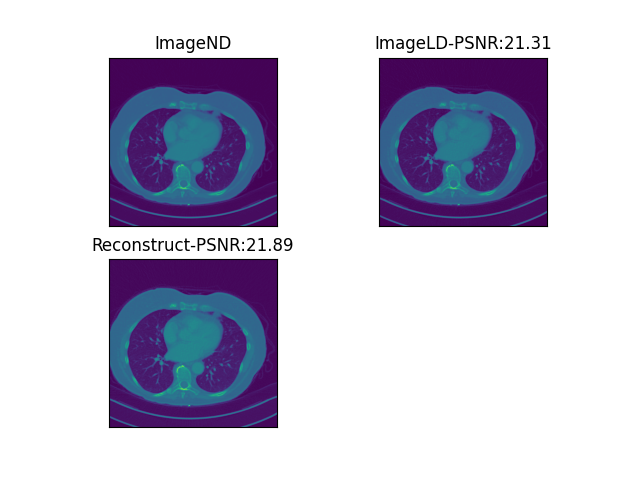
Using parameters as used in the tutorial described in **Part2** to train dictionaries on ND CT images doesn’t produce better results. Main culprit for lower PSNR score was found to be the patch size. Greater the patch size larger the learned dictionary becomes which results in learning more redundant features. Moreover, larger patch size over smooths the image which further lowers the PSNR score. Following figure shows the reconstructed image using default hyperparameters.



It was observed that dictionary learning performed worse than all denoising filters on default settings. By changing hyperparameters like number of patches to extract from each image and dictionary vector size seems to improve performance and at best comes on par with box filter but couldn’t compete with median and gaussian filter. I tried following combinations of these hyperparameters:

|  |  |
| --- | --- |
| Hyperparameters | Output file |
| Max\_patches=1000  Number of dictionaries = 100 | Results\_1000mp\_100nd.csv |
| Max\_patches=1000  Number of dictionaries = 200 | Results\_1000mp\_200nd.csv |
| Max\_patches=5000  Number of dictionaries = 500 | Results\_5000mp\_500nd.csv |
| Max\_patches=10000  Number of dictionaries = 1000 | Results\_10000mp\_1000nd.csv |
| Max\_patches=20000  Number of dictionaries = 200 | Results\_20000mp\_200nd.csv |

###### *Custom Parameters:*

When patch size of (3x3) was used, effective dictionaries were learnt that helped to improve the PSNR above the baseline. Following figure shows the reconstructed image using smaller patch size (3,3). I tried a few combinations of hyperparameters which are stated in the table below:

|  |  |
| --- | --- |
| Hyperparameters | Output file |
| Patch\_Size = (3,3)  Max\_patches=1000  Number of dictionaries = 100 | part3\_partial\_fit(3x3 1000mp 100nc).csv |
| Patch\_Size = (3,3)  Max\_patches=1000  Number of dictionaries = 50 | part3\_partial\_fit(3x3 1000mp 50nc).csv |
| Patch\_Size = (5,5)  Max\_patches=1000  Number of dictionaries = 100 | part3\_partial\_fit(5x5 1000mp 100nc).csv |

Results are provided in the CSV file containing several columns which are as follows:

* **LD\_name:** LDCT image name
* **ND-LD\_psnr**: PSNR score between ND and LD CT image. This score will be used as a baseline.
* **Box\_psnr:** PSNR score between ND and Box filter applied on LD image.
* **gaus\_psnr:** PSNR score between ND and Gaussian filter applied on LD image.
* **sharp\_psnr:** PSNR score between ND and Sharpness filter applied on LD image.
* **med\_psnr:** PSNR score between ND and Median filter applied on LD image.
* **NDvsLD:** same as ND-LD\_psnr above (baseline score).
* **Dict\_PSNR:** PSNR score between ND and reconstructed LD image using dictionary.