Master's thesis

Denoising of X-ray and electron microscopy images

submitted by

Anil Kumar, Mysore Badarinarayana

Master's of Computational Logic (MCL)
Faculty of Computer Science
Technical University of Dresden

External Advisor:
Dr. Christoph Pratsch

Academic Supervisor:
Dr. Florian Jug

Submitted February 03, 2022

Declaration of authorship

I hereby declare that I wrote this thesis with the title *Denoising of X-ray and electron microscopy images* and I used only the references and auxiliary means indicated in the thesis.

Dresden, February 03, 2022

Anil Kumar, Mysore Badarinarayana

Abstract

In X-ray and electron microscopy, obtaining a noise-free image is often difficult while examining biological samples or delicate materials. Therefore, image denoising is an essential process for the analysis of such noisy images. State of the art image denoising methods are dominated by supervised CNN based methods. However, if a noise-free ground truth is unavailable, it is not possible to use supervised CNNs. To address this problem, a denoising algorithm is proposed that uses self-similarity in images. The proposed method does not require noise-free images for the denoising task. It is based on the idea that averaging images with the same signal having independent noise suppresses the overall noise. In order to evaluate the performance of this new method, successful denoising methods that do not require a noise-free image were applied on the experimental images and the results were compared with the proposed algorithm. The results of proposed algorithm show promise in denoising images that especially have similar regions within them. The run-time performance of the algorithm is analyzed and optimizations are made to improve the run-time efficiency. The proposed method is extended to use multi-modal images where image information from different modes can be shared for the denoising process.

Acknowledgment

I would like to thank my supervisor Dr. Christoph Pratsch for constantly guiding and supporting me. I appreciate his contributions of time and ideas towards this thesis. Working with him has made me a better researcher. I would also like to thank Dr. Florian Jug for his valuable feedback, which always helped me to improve the way I work. My gratitude also goes to Prof. Dr. Ehrenfried Zschech and Prof. Dr. Gerd Schneider. Without them the collaboration with Helmholtz Zentrum Berlin would not have been possible. Lastly, I thank my family and friends for their encouragement.

Contents

List of Figures **4**

List of Acronyms **7**

1 Introduction **9**

2 Background **12**

2.1 Transmission Electron Microscopy (TEM) 12

2.1.1 Imaging modes 13

2.1.2 Scanning Transmission Electron Microscopy (STEM) . 14

2.2 Noise in images 14

2.3 Metrics and methods for evaluating denoised images 16

2.3.1 Mean Squared Error (MSE) 17

2.3.2 Peak Signal-to-Noise Ratio (PSNR) 17

2.3.3 Structural Similarity Index Metric (SSIM) 17

2.3.4 Fourier Image Analysis 18

2.3.5 Fourier Ring Correlation (FRC) 20

2.3.6 Visual Assessment 21

3 Image Denoising **22**

3.1 Spatial domain methods 22

3.1.1 Linear methods 23

3.1.2 Non-Linear methods 23

3.2 Transform domain methods 24

3.2.1 Fourier transform 24

3.2.2 Block Matching and 3D filtering (BM3D) 24
3.3 Convolutional Neural Network (CNN) based denoising methods 25
3.3.1 Supervised learning 25
3.3.2 Unsupervised learning 27
3.3.3 Self-Supervised learning 27

4 Literature Overview 28

5 Experimental data 31

6 Implementation - Application of denoising techniques 33
6.1 Generic auto-encoder for image denoising 34
6.1.1 Results 35
6.2 Block Matching and 3D filtering (BM3D) 36
6.2.1 Algorithm 37
6.2.2 Results 38
6.3 Noise2Void (N2V) 39
6.3.1 Training 41
6.3.2 Results 41
6.4 Summary 43

7 Non-Local Self-Similarity based image denoising 44
7.1 Outline of the algorithm 46
7.2 Parameters of the algorithm and stability 54
7.3 Analysis of the results 58
7.4 Comparison of results 67
7.5 Results on natural images 69
7.5.1 Comparison of the results using natural images 70
7.6 Run-time performance of the algorithm 72
7.6.1 Time Complexity of the algorithm 73
7.6.2 Slow sections of the algorithm 74

8 Optimization for run-time performance and denoising of multi-modal images 75
8.1 Optimization for run-time performance 75

8.1.1 Parallelization 75
8.1.2 Cosine similarity to improve run-time performance . . 78
8.2 Denoising of multi-modal images 80

9 Future Work and Summary **82**
9.1 Future Work 82
9.1.1 Application of transfer learning 82
9.1.2 Ensemble denoising 83
9.2 Summary 84

A Algorithms **85**

B Other results **88**

List of Figures

2.1 Schematic diagram of a TEM setup 13
2.2 Dark and bright field images 14
2.3 Example: Noisy image and its FFT 20
2.4 Example: Clean image and its FFT 20

5.1 Experimental data 31

6.1 Architecture of the auto-encoder 34
6.2 Generic auto-encoder results 36
6.3 BM3D results 38
6.4 N2V results 41

7.1 Non-Local Means result 46
7.2 Visualization of how reference patches are formed 47
7.3 Flowchart of the algorithm 48
7.4 Initial *reference patches* 48
7.5 Template matching example 49
7.6 Final *reference patches* 51
7.7 Algorithm result before the clustering step 52
7.8 Non-Local Self-Similarity based image denoising result 53
7.9 Variation of computation time with changes in the patch size 55
7.10 Effect of clustering with very few clusters 57
7.11 Effect of clustering with too many clusters 58
7.12 Non-Local Self-Similarity based image denoising result 59

7.13 FFT of the Non-Local Self-Similarity based image denoising result . . . 59
7.14 Suppression of random noise v/s the number of frames averaged 60
7.15 Visualization of the similar patches in the image . . . 61
7.16 Clustering - good example . . . 62
7.17 Clustering - bad example . . . 63
7.18 Visualization of the number of patches averaged at every pixel location . . . 64
7.19 Centroid and variances of clusters . . . 65
7.20 Denoised image and it's confidence map . . . 66
7.21 Non-Local Self-Similarity based image denoising result for all experimental data . . . 66
7.22 Comparison of results obtained from different methods . . . 68
7.23 Non-Local Self-Similarity based image denoising result on a normal image . . . 69
7.24 FFT of Non-Local Self-Similarity based image denoising result on the normal image . . . 69
7.25 Comparison of natural image results . . . 71
7.26 Run time v/s input size - denoising algorithm . . . 73

8.1 First image from the stack and its denoised result. Example of multi-slice denoising . . . 77
8.2 Comparison of the results from template matching and cosine similarity . . . 79
8.3 Results on running multi-modal images . . . 81

Acronyms

TEM Transmission Electron Microscopy 2

STEM Scanning Transmission Electron Microscopy 2

MSE Mean Squared Error 2

PSNR Peak Signal-to-Noise Ratio 2

SSIM Structural Similarity Index Metric 2

FRC Fourier Ring Correlation 2

N2V Noise2Void 3

CNN Convolutional Neural Network 3

BM3D Block Matching and 3D filtering 3

FFT Fast Fourier Transform 18

NLM Non Local Means 23

Chapter 1

Introduction

Microscopes are used to acquire highly magnified images, which are helpful in solving numerous problems in different fields including, biomedical science, chemistry, technology and industry. However, at times the noise in the acquired images corrupt the signal beyond a useful level. This leads to the problem of image denoising, which is a computer vision problem where the underlying goal is to estimate the clean signal from an image that is corrupted by noise. Noise in images can appear due to intrinsic reasons like problems in the sensors or the digital circuits, or due to external factors like the environment. Shot noise, which is a type of intrinsic noise is statistically unavoidable. The presence of noise in images deteriorate the signal quality. This deterioration can make it hard for the observer to interpret the information in images. Image denoising plays a vital role especially in denoising microscopy images since capturing good quality images is often difficult.

X-ray and electron microscopy are popular methods for the analysis of materials and structures at molecular and atomic levels. High energy X-ray and electron beams interact with the specimen and can be used to produce an enlarged image of the specimen. As these beams interact with the specimen, some rays are absorbed by the specimen. The quantity of radiation absorbed by the specimen in unit time is called dose rate.

For obtaining high resolution images of the specimen at an atomic level, the specimen has to be exposed to the radiation for a long period of time.

Hence, high resolution images are associated with high dose rates. Often, the specimen under examination consists of biological samples or delicate materials. A high dose rate on such samples can destroy the specimen under examination. Hence, there is a trade-off between the dose rate and the resolution of the image.

To preserve the structure of the specimen, it is often desirable to use low dose rates in X-ray and electron microscopy. Low dose rate imaging is similar to natural photography in low lighting conditions. Therefore, low dose rate results in a low signal-to-noise ratio, thus resulting in a noisy image of the specimen. These noisy images are not always useful for further analysis, as the signal-to-noise ratio is low and hence the information in the image is hard to interpret. The *Rose criterion* [Ros77] states that a signal-to-noise ratio of at least 5 is needed to be able to distinguish image features with certainty. Therefore after image acquisition, numerical processing is required to enhance the quality of the image so that it can be used for further analysis.

Various image denoising algorithms were proposed in the past. Conventional denoising methods like those discussed in [BCM11] and [DFKE07] use only noisy images for the denoising task, where as most modern methods involving a deep neural network like those discussed in [ZZZ18] and [ZZC$^+$17] require clean images as ground truths for training. Since clean X-ray and electron microscopy images are not available in most cases, modern denoising methods that require clean images as ground truths cannot be used. However, some deep neural network based methods like those discussed in [LMH$^+$18] and [KBJ19] use noisy supervision and self supervision respectively. These methods have shown success in denoising images without ground truth images.

The goal of this thesis is to denoise X-ray and electron microscopy images. To achieve this goal, various denoising techniques are studied and the methods that are suitable for the problem at hand are tested on the experimental images. A new denoising algorithm is also proposed and its results are analyzed and compared with other denoising methods. The proposed algorithm is then optimized to improve the run-time performance. Finally

the proposed method is improved to denoise multi-modal images.

In the following chapters, the required background knowledge and the pre-requisite concepts of image denoising are explained in brief. Next, a short review about related methods and the most relevant literature are discussed. This is followed by the description of the experimental data and the implementation of existing denoising algorithms. The proposed method is explained, its results are analyzed and the optimization methods are discussed in the next chapter. Finally the potential improvements of the algorithm are outlined and the results of the thesis is summarized.

Chapter 2

Background

2.1 Transmission Electron Microscopy (TEM)

Transmission Electron Microscopy (TEM) is a microscopy method in which a beam of electrons is transmitted through a specimen. The schematic setup of a TEM is shown in figure 2.1.

In TEM, an electron beam with a uniform current density is made to interact with the specimen. This beam is emitted from the electron gun with an acceleration voltage, typically in the range of 80kV to 300kV. A TEM includes a condenser lens system responsible for varying the electron intensity distribution and the area of the illuminated specimen. The electron-intensity distribution behind the specimen is imaged with a multi-stage lens system onto a fluorescent screen or a direct electron detector [Rei84].

The interaction of electrons with the specimen results in elastic and inelastic scattering. Multiple interactions result in a decrease of the resolution and lead to significant problems in the interpretation of images. In order to prevent multiple interactions, the specimen must be very thin, typically less than 100nm for 100keV, depending on the specimen properties and the resolution required [Rei84].

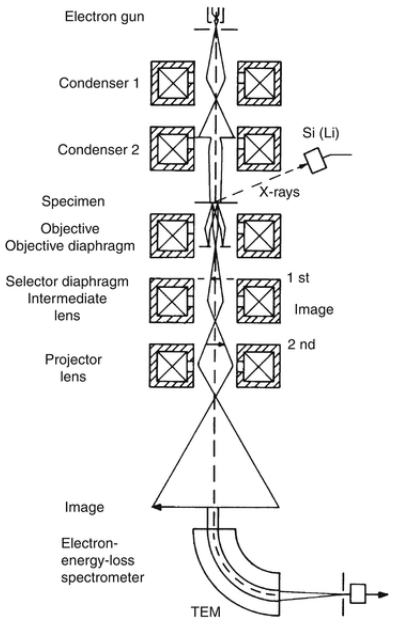

Figure 2.1: Schematic diagram of a Transmission Electron Microscopy (TEM) setup [Rei84]

2.1.1 Imaging modes

There are several imaging modes in TEM. However, only the images obtained in bright-field and dark-field mode are used in this thesis. Bright-field mode is similar to the imaging in a classical optical microscope. The natural scattering of the specimen can be observed in bright-field mode. The specimen appears dark, and the background appears bright in this mode. The contrast can be adjusted by adjusting the condenser. However, this affects the resolution of the image [Kim13].

On the other hand, specimen appears brighter on dark background in dark-field mode. A bright-field microscope can capture images in dark-field mode by blocking the direct beam of the source hitting the detector. The specimen appears bright as they deflect the source beams and are detected by the objective. Dark-field microscopy can capture minute structures [Kim13].

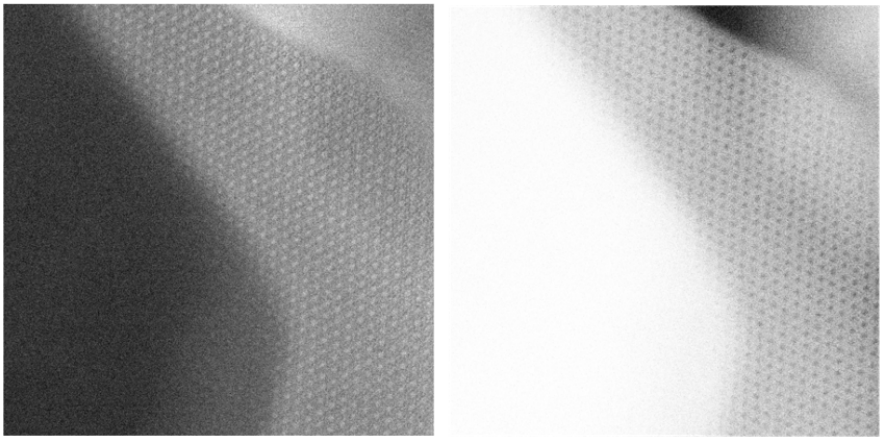

Figure 2.2: Image of the same specimen in bright-field contrast(left) and dark-field contrast(right)

2.1.2 Scanning Transmission Electron Microscopy (STEM)

A conventional TEM can be used for STEM with additional scanning coils, detectors, and circuits. In STEM, the electron beam is focused to a single spot (typically of the order 0.05-0.2nm) and is made to scan each point on the surface of the specimen with the beam parallel to the optical axis [Rei84]. The main advantage of STEM is that different imaging modes can be used at the same time. Figure 2.2 is an example of STEM image, where bright-field and dark-field images of the same specimen are shown (Note: The background in these images can be seen in the top right corner. All other regions represent the structure).

2.2 Noise in images

Image noise is the random variation in brightness, intensity, or color of an image. The corruption of pixel values deteriorates the signal and results in images that are poor in quality. There is some level of noise in most images which is undesirable [Hos07]. The main causes for image noise are:

- problems in image sensors,

- the circuitry of digital cameras,
- photon detectors,
- data transmission error,
- properties of imaging systems such as air turbulence, water droplets or poor lighting,
- image acquisition errors, etc.

There are various types of image noise. Some of the major ones are as follows:

- **Fixed Pattern Noise**: An image sensor may consist of a series of detectors, where individual detectors do not have identical responses. In such cases, noise is present in images in an additive or multiplicative way [Hos07]. Additive noise can be represented as:

$$y(x, y) = s(x, y) + n(x, y) \tag{2.1}$$

  where $s(x, y)$ is the noise-free signal, $n(x, y)$ is the additive noise, and $y(x, y)$ is the detected image. In case of multiplicative noise, $y(x, y)$ which consists of a multiplicative factor $p(x, y)$, is represented as:

$$y(x, y) = p(x, y) * s(x, y) + n(x, y) \tag{2.2}$$

- **Gaussian noise**: This type of noise is produced by sensors during image acquisition. Factors like the illumination intensity, sensor temperature, electronic circuit associated with the sensor, etc., contribute to Gaussian noise [Cat12]. Typically, this type of noise is additive, independent at each pixel, independent of the signal intensity, and follows a Gaussian distribution.
- **Shot Noise**: All imaging systems measure the radiation of some sort to capture images. It can be optical intensity, electron current, radio power, X-ray counts, etc. Since all these sources are discrete, the radiation at the detector is measured by counting the number of particles

incident on it. This can be considered as a discrete event. During the imaging process, a series of such discrete events occurs. The average time interval between two particles getting detected can be calculated, but the time interval between independent particles is random. Consequently, the number of particles detected at a given interval also varies [Hos07].

The variation in the number of particles detected at different intervals results in shot noise. This type of noise can be modeled be a Poisson's process. Shot noise is dominant in the brighter parts of the image. This type of noise is proportional to the square root of the signal intensity, but the noise at different pixel locations is independent [Mac06].

- **Salt and pepper noise**: This noise is usually created during data transmission or by analog to digital converters. The noise is a result of bit errors during transmission. Images with salt and pepper noise have dark pixels in bright regions and bright pixels in dark regions [GW06].

- **Quantization noise**: This noise is caused by quantizing the pixel values to discreet levels. When pixel values are transformed from the actual values, noise is introduced.

- **Periodic noise**: An electrical or electromechanical interference results in periodic noise during imaging. This type of noise can be identified by repeated patterns that are added on top of the actual image.

2.3 Metrics and methods for evaluating denoised images

Image noise can be suppressed by applying denoising methods on noisy images. For evaluating denoised images, specific metrics and techniques are used. These are briefly described in the following.

2.3.1 Mean Squared Error (MSE)

The deviation of the denoised image from the original noise-free image determines the quality of denoising. In MSE, the square of the difference between each pixel value from the clean image and the denoised image is found and then averaged over space [Kna13]. Let x be the noise-free image, x' be the denoised image, and p be the different pixels in these images. Then MSE is given by:

$$MSE = \frac{1}{wh} \sum_{p} (x - x')^2, \tag{2.3}$$

where w and h are the width and height of the images. wh denotes the total number of pixels in the image.

2.3.2 Peak Signal-to-Noise Ratio (PSNR)

PSNR is based on MSE. In simple words, PSNR is the logarithmic measure of normalized MSE. Let x_{peak} be the maximum possible pixel value, then PSNR is represented as,

$$PSNR = -10\log_{10} \frac{MSE}{x_{peak}^2} \tag{2.4}$$

PSNR is a measure represented in decibels(dB). Since PSNR is a normalized value, it is more useful, and hence this is one of the most popular metrics for evaluating denoised images. Typically, for a 8-bit data representation, PSNR values vary between 30-50dB [Kna13][SAU19].

2.3.3 Structural Similarity Index Metric (SSIM)

SSIM is defined over a neighborhood of a given pixel. For two images, x and y, SSIM is calculated as,

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)}{(\mu_x^2 + \mu_y^2 + c_1)} \frac{(2\sigma_{xy} + c_2)}{(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.5}$$

where μ_x and μ_y are the means, σ_x^2 and σ_y^2 are the variances and σ_{xy} is the covariance of x and y calculated for a pixel with a fixed neighborhood.

The constants, c_1 and c_2 are used to stabilize the division with weak denominators. To compute SSIM of the image, SSIM is averaged over all pixel positions [Kna13].

SSIM is a perception-based metric. The change in structural information is considered to measure image degradation. Lately, SSIM is becoming a popular method to evaluate denoised images [WBSS04].

2.3.4 Fourier Image Analysis

Even though the metrics mentioned above are well established for evaluating denoised images, these metrics require data from the original noise-free images. Hence, these metrics cannot be used in the absence of an original noise-free image. In such cases, Fourier image analysis is helpful.

The Fourier transform converts data from the spatial domain into the frequency domain. Data like audio signals are easily interpreted in the frequency domain than in the time domain. Hence, Fourier transform is popularly used in processing and analyzing audio data. On the other hand, it is not easy to understand the information when images are transformed into the frequency domain. Images are easily interpreted in spatial domain [Ste97].

Converting images in the spatial domain to the frequency domain results in an array with complex numbers, i.e., data representing both phase and magnitude. The Fourier transform having size $N * N$ can be thought to have four quadrants with each quadrant of size $N/2 * N/2$. The upper right corner is a mirror of the lower left. Similarly, the upper left corner is a mirror of the lower right. This symmetry holds for both magnitude and phase data. Additionally, in the Fourier transform of an image, the amplitudes with lower frequency are placed at the corner, and those with higher frequency are present at the center. Even though nothing much can be directly inferred from the Fourier transform, image features are represented by some structures in the Fourier transform [Ste97].

Fourier transform of images can be computed in Python by using Fast

Fourier Transform (FFT)[1] from the numpy library. Since the Fourier transform of images has the zero frequency at the corner of the computed numpy result, it is shifted to the center using *fftshift* [2] function of the numpy library, after computing the FFT. So when referring to Fourier transform of images in the future, the zero frequency is considered to be at the center.

The Fourier transform of an image can be seen in figures 2.3 and 2.4. The structures in the Fourier transform represent the information in the images. Periodic noise in images can be removed by applying filtering techniques on the Fourier spectrum, but removing non-periodic noise using similar techniques is difficult [Lar11].

Evaluating a denoised image using Fourier transform can be challenging as it is not straightforward to interpret. The Fourier transform of a noisy image has to be compared with the Fourier transform of its denoised image to identify structural enhancement or suppression in the noise.

Figure 2.3 shows an image with salt and pepper noise and its corresponding Fourier transform. Figure 2.4 shows the same noise-free image and its corresponding Fourier transform. From both the Fourier transforms, one can observe that the corners of the Fourier transform in figure 2.4 is darker than its counterpart in figure 2.3. These dark corners represent the suppression of the noise, which can be verified by observing the images in spatial domains in figures 2.3 and 2.4. These images are examples of how denoised images are evaluated using Fourier transforms.

[1] https://numpy.org/doc/stable/reference/generated/numpy.fft.fft2.html
[2] https://numpy.org/doc/stable/reference/generated/numpy.fft.fftshift.html

Figure 2.3: Image with salt and pepper noise(left) and it's Fourier transform(right) [1]

Figure 2.4: Noise free image(left) and it's Fourier transform(right) [1]

2.3.5 Fourier Ring Correlation (FRC)

FRC is being used since decades to estimate image resolution, especially in electron cryomicroscopy [KTC+19]. It is based on a normalized cross-correlation histogram measure which is calculated in frequency domain. FRC is used for analyzing two images of the same region of interest having independent noise realizations [KTC+19]. FRC histogram is calculated by the following equation.

[1] https://www.ismll.uni-hildesheim.de/lehre/ip-14s/script/imageanalysis-04-fourier-transform-2up.pdf

$$FRC(r_i) = \frac{\sum_{r \in r_i} F_1(r) \cdot F_2(r)^*}{\sqrt{\sum_{r \in r_i} F_1^2(r) \cdot \sum_{r \in r_i} F_2^2(r)}}, \tag{2.6}$$

where F_1 and F_2 are the Fourier transform of two images, r_i is the i^{th} bin frequency (the frequency spectra are divided into bins) and $*$ represents the complex conjugate. The histogram is used to define the image resolution and a cut-off frequency is identified at which the correlation of the images drop significantly. FRC is quantitative and depends on the sample and microscope characteristics [KTC+19]. However some of the limitations of FRC is mentioned in [JBC+21] which points out that FRC can be biased to classify lower resolution images as better resolved.

2.3.6 Visual Assessment

The metrics mentioned above might not be a good representation of the result under certain circumstances. MSE is just a measure of how much the denoised image differs from the original noise-free image. The work of [PN15] shows the limitations of SSIM in medical imaging. Fourier transform, as mentioned earlier, is not easy to interpret. Sometimes, a human eye can be better at perceiving the correctness of a denoising task. Hence, the human visual system is also used to evaluate how well an image has been denoised [Kna13].

Chapter 3

Image Denoising

Noise can corrupt the image quality beyond a useful level. Noise can exist due to external factors like lighting quality, the number of photons or electrons hitting the image sensors, or internal factors like sensor problems, analog signal conversion to a digital signal, to name a few. Hence researchers have come up with data processing methods to improve the signal quality of the image after acquisition.

Image denoising algorithms can be broadly classified based on the approach being used for denoising as follows [FZFZ19] :

- Spatial domain methods.
- Transform domain methods.
- Convolutional Neural Network (CNN) based denoising methods.

3.1 Spatial domain methods

In spatial denoising methods, the general idea is that the spatially available information around a position (x,y) is used to calculate the new pixel value at (x,y). Linear, non-linear, non-local regularization, total variation regularization, sparse representation, and low-rank minimization are some spatial domain methods [FZFZ19]. Among these, the popular methods are briefly explained below.

3.1.1 Linear methods

Linear methods contain linear coefficients convoluted along the image's different positions to obtain a denoised image. Mean filtering, Gaussian filtering, and Wiener filtering are some examples of linear filtering. The denoised images obtained from applying linear filtering have a smoothing effect, and thus the information around sharp edges is lost[FZFZ19]. These days, linear filtering is usually employed in more complex denoising methods rather than using it independently.

3.1.2 Non-Linear methods

Non-linear methods often perform better than linear methods. For example, they help suppress non-Gaussian noise (like spikes) and have edge-preserving properties. Median filtering, bilateral filtering and non-local means [BCM11] are some of the most commonly used non-linear methods.

Median Filtering

In median filtering [mDDS+15], a patch in an image with $2N + 1$ values is selected, and the values are sorted. The middle value is the median of the sample. This value is used to replace the center value of the patch in the image. Median filtering is known for the preservation of edges during noise removal [mDDS+15].

Bilateral Filtering

Bilateral filtering [TM98] replaces the center pixel of an image patch with a weighted average of the surrounding pixels but has an additional weighting factor. The addition of this extra factor makes this method non-linear. Bilateral filtering preserves edges and creates a smoothing effect.

Non Local Means (NLM)

NLM [BCM11] is a denoising method that uses similar patches in an image to minimize the noise. In this method, patches are replaced by the average

of the similar patches in the nearby surrounding. The idea behind this method is that when n patches are averaged, the noise will be suppressed by $\sqrt{n}$ times [BCM11]. Sometimes in NLM, a weighted average is performed, and the less similar patches are assigned lesser weights while averaging is performed.

3.2 Transform domain methods

In transform domain methods, data of the image to be denoised in converted to another domain by applying transformations like Fourier transform [GRP15], cosine transform [EP16] and wavelet domain methods [GRP15] to name a few. Once the data is converted to another domain, data is processed, and then the processed data is converted back to the spatial domain to obtain a denoised image.

Fourier transform and BM3D [MAF20], which are the popular transform domain methods are explained briefly below.

3.2.1 Fourier transform

Applying Fourier transform [GRP15] on an image converts the spatial domain data to the frequency domain. Noise, which corresponds to the higher spectrum frequency, can be suppressed by passing the data in the frequency domain through a low pass filter. Applying a low pass filter on the Fourier spectrum smoothes the image. On the other hand, high pass filters make the images sharper. Restricting a part of the frequency spectrum is a primitive denoising method, which might not be sufficient to obtain good quality denoised images, especially when the type of noise involved is complex (i.e., spread across different frequencies).

3.2.2 Block Matching and 3D filtering (BM3D)

BM3D [DFKE07] is one of the popular methods that use the transform-domain denoising approach. It can be considered as an extension of non-local means [BCM11] method. In BM3D, similar patches in an image are stacked

to form $3D$ blocks. These $3D$ blocks are transformed into another domain, and collaborative filtering is applied. Later, an inverse transform is applied to convert the data into the original domain. The denoised image is then constructed from these patches. BM3D generally works well but struggles as the noise level increases [FZFZ19]. The latest version of BM3D [MAF20] has been used on the experimental data, and the method is discussed in detail in the later chapters.

3.3 Convolutional Neural Network (CNN) based denoising methods

In recent years, deep learning is being popularly used in many applications with great success. Convolution neural networks are often used to process multidimensional data like images. Hence CNN techniques are also successfully used for denoising images [KBJ19] [GAAB18] [HY18].

Deep learning or machine learning methods can be broadly classified into two types - supervised learning, which deals with the labeled data containing expected ground truths and unsupervised learning, which deals with unlabeled data having no ground truths. Additionally, there is self-supervised learning, where labels are created from unsupervised data.

3.3.1 Supervised learning

Supervised learning accounts for a significant part of the machine learning problems. Supervised learning deals with a lot of annotated training data i.e., inputs associated with outputs. Algorithms create models by learning information from the training data. These models are then used to predict the results of the new unseen unlabeled data [CCD08].

Deep neural network is an extension of machine learning that is been used widely in recent times. It consists of neurons, which can be considered as computational units. Neurons accept an input, do some processing and produce an output. Neurons consist of weights, W and bias, b [Sin21]. Other than neurons, a neural network also includes activation functions, like

sigmoid, tanh, ReLU, etc [OK11] [MDKF08] [Fan00]. These activation functions introduce non-linearity to the linear output produced by the neurons [TFZ+20]. The predicted output from a neuron, Z is represented as,

$$Z = f(W^T X + b) \tag{3.1}$$

where W^T is the transpose of the weight matrix, b is the bias and $f()$ represents an activation function. Equation 3.1 is an output of a single neuron. Deep neural networks consist of multiple neurons, often in thousands connected in a specific way. Training a deep neural network involves tuning the weight and bias variables of the neurons.

CNNs work on the same basic principle as deep neural networks, but the weights and the data processed by a CNN is often multi-dimensional. Kernels or filters, which are trainable, act as weights in CNNs [GBC16]. Training these kernels and fine-tuning them to fit the training data creates a model. This model can then be used for new unseen data.

CNNs have shown to work well for image denoising [KBJ19] [ZZC+17]. Depending on the requirements and the images being processed, the architecture of CNNs vary. For instance, images from a regular camera, X-ray images, microscopy images, etc., have different properties, and hence the data has to be processed accordingly. Therefore, there are multiple architectures for image denoising or any other task involving neural networks, and the techniques used are not fixed.

The review paper [TFZ+20] gives an overview of different types of CNN-based methods that are used for different types of data and different types of noise. In [JJR+19], CNN with Unet [RFB15], which is a popular CNN architecture, has been used for denoising medical images. For low-dose CT image denoising, [GAAB18] used CNNs with dilated convolutions [Tsa18]. [HY18] used CNN with skip connections [Ada20] for CT image reconstruction. These examples show the various configurations of CNNs tailored to cater to the problem at hand. It should be noted that all these methods require training data that is labeled.

3.3.2 Unsupervised learning

Unsupervised learning deals with unlabeled data. The training samples are used to find patterns in the data [TFZ+20]. [LYC+17] uses unsupervised image generation from key local patches for image synthesis. [YLZ+18] is another example of unsupervised image denoising.

3.3.3 Self-Supervised learning

In self-supervised learning, labels are learned from the unlabeled sample data. Pseudo labels are generated from the training data, and these labels are used to tune the weights of a machine learning network [ZOKB19]. Self-supervised learning closely resembles the way humans classify objects [Bou20].

Annotating large datasets is often a laborious manual task, and some tasks like image denoising may not have ground truth data. Since supervised learning does not require annotated data or ground truths, self-supervised learning is becoming increasingly popular [Sus].

Self-supervised learning has been successfully used for image denoising tasks. In Noise2Void (N2V) [KBJ19], the image is divided into multiple patches. The center pixel of a patch is predicted from its surrounding pixel values. Here, the center pixel is labeled as output, and the surrounding pixels are labeled as input from the unlabeled image. Noise2Inverse [HPB20] and [LKLA19] are some of the other popular self-supervised denoising methods.

Chapter 4

Literature Overview

The image denoising problem is being studied for around 50 years [TFZ+20]. The earliest methods mostly consisted of non-linear and non-adaptive filters [HUA71]. As time progressed, more filtering methods were developed. Weighted median filtering [YYG+95], median filtering [GW+02] [PV13], bilateral filtering [TM98] are some of the examples of filtering methods. Later there was a shift from spatial methods to transform domain methods. In the last two decades, wavelet transform [GRP15] has gained popularity [MGMH04]. The methods discussed in [Hou07], [JHWL08] and [ZBW05] are some of the examples of denoising techniques that used wavelet transform.

At the same time, new spatial domain methods too were developed. One such method is Non Local Means (NLM). NLM [BCM05] is a technique that uses similarity within images and applies filtering techniques. It was very successful when it was introduced. The improvised version of NLM are still being used these days. Inspired by NLM, Block Matching and 3D filtering (BM3D) [DFKE07] was introduced. BM3D used the ideas of NLM to identify similar regions of an image and processed these similar regions in a transform domain to obtain the denoised image. BM3D is one of the classical denoising methods and it is being used successfully even today. The initial success of BM3D was a motivation for researchers to develop many improved versions of it. The current version of BM3D was published in 2020 [MAF20].

Apart from spatial and transform domain methods, machine learning methods are also used for image denoising. Early examples from the 80s and 90s include [CS89] and [BT92]. With the availability of powerful computing devices, machine learning methods gained a lot of popularity in the last decade. Several state of the art denoising methods are based on deep neural networks. Convolutional Neural Network (CNN) based methods like the ones proposed in [ZZC$^+$17] and [ZZZ18] have been successful in producing good quality results. In [ZZC$^+$17], residual learning is used where the network tries to learn the noise structure, which is then removed from the noisy image to get a denoised output. [ZZZ18] combined the benefits of [ZZC$^+$17] with a faster computation speed and a better quality of denoising. This is an example of how neural network based methods have improved based on similar previously proposed methods.

This thesis focuses on the denoising of X-ray and electron microscopy images. Absence of noise free images means that most of the state of the art supervised learning methods cannot be used. For these type of denoising problems, methods like Noise2Noise [LMH$^+$18] can be used. In Noise2Noise, supervised learning is carried out with noisy image pairs. This means that a clean noise free image is generated with the help of noisy images only. To denoise microscopy images without a ground truth, neural network based methods were proposed in [KBJ19], [BR19], [XWJ20], [KVP$^+$20] and [PKJ21] to name a few. In [KBJ19], [BR19] and [XWJ20], self supervised learning is used and the labels are created from the information existing in the noisy input image. The CNN network used to train these methods does not have information about clean images and hence they cannot be expected to always outperform supervised learning methods. Hence to improve the quality of denoising, probabilistic techniques were proposed in [KVP$^+$20], [PKJ21] and others, which are unsupervised learning methods. Image denoising is performed by creating a probabilistic noise model in these methods.

The survey of image denoising techniques in [FZFZ19] and [TFZ$^+$20] compare spatial domain, transform domain and CNN based methods. Although there are statistical methods that are used even to date that do not

require information from a clean noise-free image, there are very few CNN based denoising methods that are successful in reliably producing good quality results without clean noise-free signals.

In this thesis a new denoising algorithm is proposed, which is primarily targeted at denoising X-ray and electron microscopy images. The algorithm uses global self similarity in images, the idea for which was inspired by Single Particle Imaging [Fra96]. In the proposed method, similar regions in the image are clustered and then averaged to remove noise. Although the methods proposed in [CM05] and [Mig07] use the concept of averaging for image denoising, they are mostly a minor extension of the NLM [BCM05]. The proposed algorithm uses global self-similarity as opposed to finding similar patches in the close vicinity in NLM. This helps in finding more similar components and averaging these components will suppress the noise further.

In this thesis, the denoising methods that suits our problem have been identified from the literature review. These methods have been applied on the experimental data and the results are analyzed. Later, a new denoising algorithm has also been proposed.

Chapter 5

Experimental data

X-ray and electron microscopy is often used for examining physical properties of samples in material science. These methods can provide insights into the structure at a nano-scale level. However, the resulting images are often extremely noisy. Therefore, the feasibility of denoising this type of experimental data was investigated. Some of the experimental images are shown in figure 5.1. These images were provided by the Department of Inorganic Chemistry, Fritz Haber Institute of the Max Planck Society. The information of the samples present in these images were not disclosed as it is confidential.

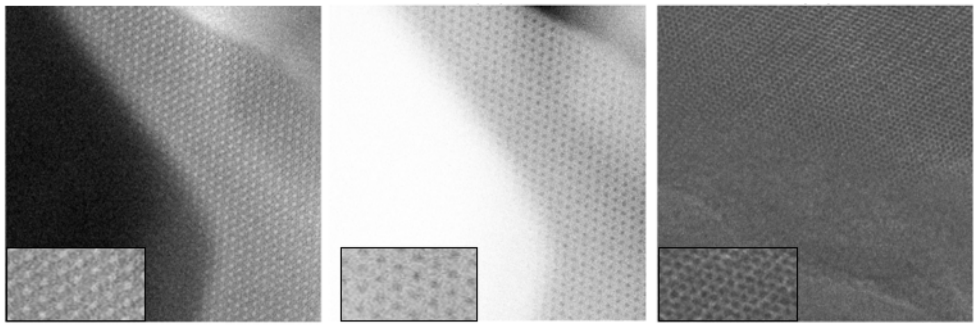

Figure 5.1: Experimental data

The leftmost and the center images in figure 5.1 contain the same sample. The leftmost image is recorded in bright field mode and the center

image is recorded in dark field mode. The rightmost image is an image of another sample. These images use a .dm3 or a .emd file extension. DM3 is a file format from Digital Micrograph by Gatan [1]. It is a raster image file format in which images from digital microscopes, scanners and cameras can be saved. A DM3 file also contains metadata information like the timestamp, camera attributes and the other parameters of the Transmission Electron Microscopy (TEM). Another image format used was the Electron Microscopy Dataset (EMD) file format, which was developed by Colin Ophus at National Center for Electron Microscopy. It is a subset of the HDF5 wrapper format and can store N-dimensional data arrays of any standard data format. These DM3 and EMD files can be read by a Python library called hyperspy [2]. It should be noted that the range of pixel values in the experimental images are in the order of 10^6 unlike most normal images where the data range is from 0 to 255.

The images are noisy and the main type of noise in these images is shot noise [MDP14] [STP04]. The intensity of the noise levels in these images are not known, which can be problematic because some denoising methods like the BM3D require information about the noise intensity for the denoising task. These images do not have equivalent noise free images too. This means that supervised CNN based methods cannot be applied on these images.

The rightmost image in figure 5.1 is a part of an image stack consisting of 10 images. The images in the stack are of the same specimen and all these images appear to be very similar, but the alignment of the data varies minutely in each image. Information from different images within such stacks is shared in the proposed denoising algorithm, which is explained in chapter 8.

Later chapters also explain how the individual images from the experimental data and how multi-modal images (dark field and bright field images in figure 5.1) are denoised.

[1] https://www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software
[2] http://hyperspy.org/hyperspy-doc/current/user_guide/intro.html

Chapter 6

Implementation - Application of denoising techniques

In microscopy, even obtaining multiple noisy images can be impossible since the specimen under examination cannot be exposed to radiation for a long duration without damaging it. A noisy image is a mixture of true signal (i.e., the image with zero noise), s and the noise component, n. Therefore a noisy image x can be denoted by,

$$x = s + n \tag{6.1}$$

Image denoising techniques aim to reduce the noise component in noisy images. Various denoising techniques have been used in the past to achieve this, some of which were discussed previously. The noise levels of the experimental images are unknown and the ground truth data is not present for these images. Considering these constraints, a few denoising techniques were chosen and applied. This chapter summarizes the results obtained from these denoising algorithms.

6.1 Generic auto-encoder for image denoising

Auto-encoders [MSY16] have been famously used for image denoising. Inspired by the application of auto-encoders in image denoising and the successful results they have produced in the past, a simple and custom autoencoder was developed to perform image denoising.

Often auto-encoders perform supervised learning. Hence in such cases, a noisy image and corresponding noise-free image pairs are required for training. Since original noise-free data is not present for the experimental data, same images were used for the input and the output. In scenarios where the same data is used as an input and an output, an auto-encoder can potentially learn an identity function. However, auto-encoders with hidden layers smaller than the input layers cannot behave as an identity function. The features size reduces as the data is passed to the smaller layers. When the data has to be reconstructed back to the original size, all the features of the input cannot be precisely reconstructed [Ben09] [Ng17]. The resulting prediction with unseen data will be slightly different even when the autoencoder is trained with the same inputs and outputs [Urb17].

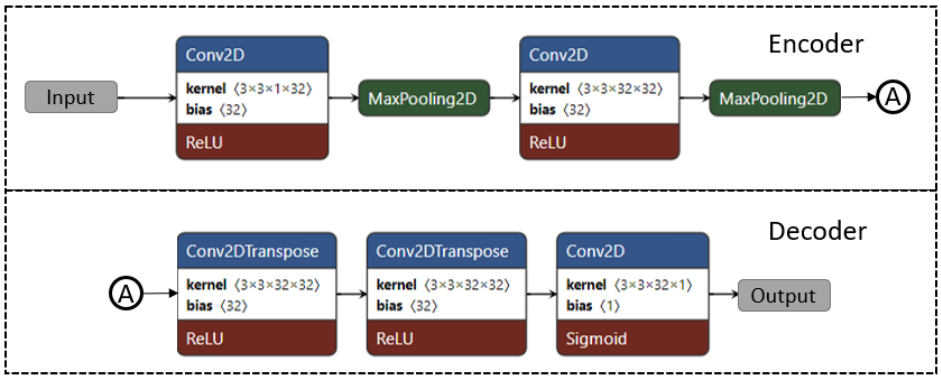

Figure 6.1: Architecture of the auto-encoder

The idea behind this auto-encoder setup is similar to that in N2V. The noise-free information has dependencies on its surroundings which can be predicted, unlike the noise, which is independent. In this implementation, too, patches of images are used. The denoised image is obtained by applying

the trained model on the noisy image.

The network is as shown in the figure 6.1. The network architecture has been taken from [Cho16] and some minor changes have been made in the configuration. This network has been popularly used to denoise the MNIST dataset [LC10]. The encoder part of the network consists of two $2D$ convolution layers, which are connected to two $2D$ MaxPooling layers as shown in the figure 6.1. Each $2D$ convolution layer uses a set of 32, $3 * 3$ kernel, the output of which is connected to a ReLU [OK11].The decoder part consists of two $2D$ convolution transpose layers that are individually connected with ReLU and have a set of 32, $3 * 3$ kernels each. The last layer is a $2D$ convolution layer with a single filter that converts the data into the shape input has. In this implementation, the image is split into various patches taken from different regions of the image. A patch size of $48 * 48$ has been used as this size is neither too small nor too big to represent different features. The network is trained with these patches to obtain a model, which is used on the noisy data.

6.1.1 Results

The results obtained with the auto-encoder network are as shown in figure 6.2. On observing the images, one can notice that there is not a great deal of denoising (refer section 7.4). The results only seem to have improved by a tiny margin. The noise has been suppressed very little in the leftmost and the center images. The features in these images have been smoothened a bit, which can be observed in the zoomed-in sections. However, the noise is still prominently present in plain sections of the denoised images. The noise has been hardly suppressed on the rightmost image, and the denoised result looks almost identical to the noisy input. Hence it can be concluded that this denoising method does a poor job in suppressing the noise.

From figure 6.2, one can observe that the denoised result is almost similar to the noisy input. This is because that the auto-encoder almost learns an identity function which is not made possible by the architecture of the network as the layer size is reduced in the encoder part [Ng17].

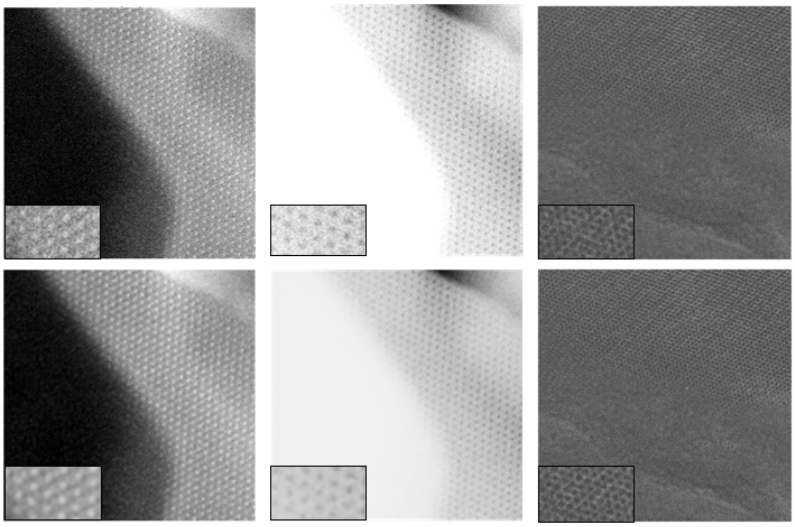

Figure 6.2: Noisy images in the top row and their corresponding denoised images in the bottom row

6.2 Block Matching and 3D filtering (BM3D)

BM3D is one of the classical denoising algorithms which is used widely for denoising images in different applications. The first version of BM3D [DFKE06] was introduced in 2006. Over time, many improvements were made on the algorithm and the current version was published in 2020 [MAF20]. BM3D is a transform domain denoising technique and it can be considered as an extension of the Non Local Means (NLM) [BCM11] method. It is a non blind denoising technique, which means that the noise level of the signal has to be known for running the algorithm on noisy images. Even though the noise levels in the data is not known, it can be approximated by finding the standard deviation of the images. From equation 6.1, it can be observed that the standard deviation of either the signal s, or the noise n, cannot be greater than the standard deviation of the noisy image x. Hence, the standard deviation of the noisy image forms an upper bound for the possible values that the standard deviation of the noise can have.

The important criterion for considering BM3D is that it is a conven-

tional method that works well without ground truth. The working of the algorithm and the results obtained on applying the algorithm is explained in the following.

6.2.1 Algorithm

BM3D is a denoising technique that uses collaborative filters [Gro17] in the transform domain. The BM3D algorithm starts with block matching. Similar patches are found in the noisy image by comparing the reference patch with other patches in the surrounding neighborhood. The similarity between patches is calculated by measuring the Euclidean distance. Similar d dimensional patches are obtained and are stacked to form $(d+1)$ dimensional data.

For two dimensional images, a $2D$ transform is applied on similar patches that are grouped. Then, a $1D$ transform is applied, which is followed by applying filters. The parameters of the filters depend on the variance, noise statistics, and other priors. In BM3D, filtering is done in two stages - Hard thresholding in the first stage and Wiener filtering in the second stage.

The computed parameters of the filters are applied to the $3D$ transformed data in a method similar to collaborative filtering [Gro17]. Then, an inverse $3D$ transform is applied to the resulting data to convert the data back into the original domain. Back in the spatial domain, this data is used to reconstruct the denoised image.

Calculation of patch variances is an essential step in BM3D. The previous versions of BM3D (for instance [DFKE07] and others) assume that the noise distribution between patches is independent. The variance calculation is approximated based on this assumption. However, noise correlation can still exist between patches which might result in artifacts. This problem is addressed in [MAF20] as an exact calculation of the variance has been employed.

It should be noted that BM3D tries to eliminate correlated noise along with independently distributed noise, which can be hard to achieve even with state-of-the-art neural network methods [MAF20].

6.2.2 Results

Since BM3D is a non-blind denoising method, it considers the noise level in the noisy images as one of its parameters. However, the noise level in the experimental data is not known. In order to estimate the noise level, the standard deviation of the images was used. BM3D algorithm was made to run with "ALL STAGES", which means that both Hard-thresholding and Weiner filtering were used during the denoising process. This mode is relatively slow but produces better results. The other mode "HARD THRESHOLDING" applies only hard thresholding, which is relatively faster. However, there is a compromise with the quality of the results obtained.

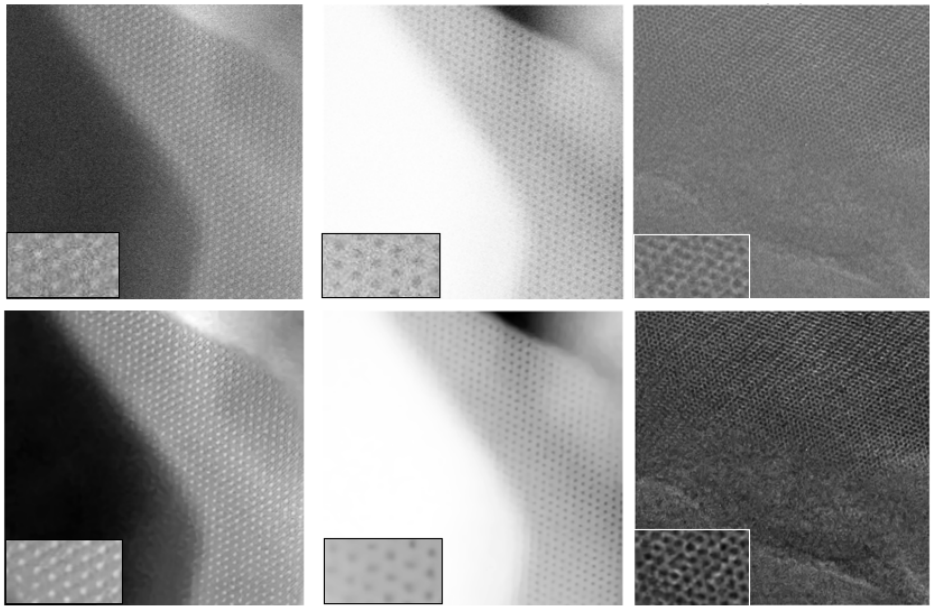

Figure 6.3: Noisy images in the top row and their corresponding denoised images in the bottom row (image-1, image-2 and image-3 from left to right)

The standard deviations of the images were initially used as the noise level parameters for BM3D, which did not provide satisfactory results. As explained above, the standard deviations of the images form the upper limit for their noise levels (i.e., standard deviations of the noise). In figure 6.3,

the standard deviations of image-1 (dark-field image) was 0.119, image-2 (brightfield image) was 0.067, and image-3 was 0.017. Different values smaller than these standard deviations were used by trial and error method to approximate the noise levels of the image. The best denoised outputs were obtained with a noise level of 0.05, 0.05, and 0.008 for image-1, image-2, and image-3, respectively. The images and the corresponding results are shown in figure 6.3.

In figure 6.3, the patches on the bottom-left of each image show the zoomed-in regions of the denoised images. It can be observed that BM3D [MAF20] did a decent denoising task, and the overall quality of the images was better than the noisy images. In the case of image-1 and image-2, the circular structures appear to be visible and have a concrete shape. However, it can be observed that the algorithm fails to reconstruct the sub-structures present in between the circular structures neatly, which is evident from the zoomed-in region of image-1. In image-3, the overall contrast of the image appears to be better. On observing the zoomed-in region, the existence of noise can still be seen. The failure to successfully reconstruct the information that is not clearly visible in original noisy data is a significant drawback.

It should be noted that BM3D has some constraints in denoising images. Transforming data from one domain to another and back might result in loss of some information [MAF20]. The loss of information can be a possible reason why the minute details are missing in the denoised image even though the denoising performance is relatively good.

6.3 Noise2Void (N2V)

While exploring the state-of-the-art denoising methods, it was observed that neural networks have been ruling the domain. Noise2Void (N2V) [KBJ19] which has proven to be successful, was studied and applied to the experimental images. The denoising technique and the analysis of the results obtained have been summarized in the following.

N2V is a self-supervised denoising method. It uses a neural network that can be trained with a single noisy image, which is beneficial, especially for

images obtained from microscopy.

As mentioned before, a noisy image is a mixture of a true signal (i.e., the image with zero noise) and the noise component. Therefore a noisy image x can be denoted by $x = s + n$, where s is the signal and n is the noise component. For denoising an image with N2V, two main assumptions are made. They are:

- The signal s is not pixel-wise independent,i.e., the value of any pixel in an image is dependent on the surrounding pixel values in its proximity.
- The noise component is conditionally pixel-wise independent given the signal s, which means that the noise intensity is not dependent on the surrounding pixels. Furthermore, the noise is assumed to have zero mean.

N2V trains the network using image patches. The patch's center pixel is considered as the output and is predicted from the other pixels in the patch. The other pixels in the patch are thus considered as the input. This idea is based on the first assumption described above. Since the signal is dependent on its surroundings, the signal component of the center pixel is predicted correctly. The noise, which is assumed to be independent, is suppressed in the denoised result. Surrounding pixels that influence the value of the center pixel are called the *receptive field*. The neural network parameter denoted by θ is tuned to minimize the pixel-wise loss. The neural network function can be defined as,

$$f(x_{RF(i)}; \theta) = \hat{s}_i, \tag{6.2}$$

where $x_{RF(i)}$ is the *receptive field* and $\hat{s}_i$ is predicted center pixel at location i. If s_i is the signal present at location i, then the loss function is calculated as follows,

$$L(\hat{s}_i, s_i) = (\hat{s}_i - s_i)^2 \tag{6.3}$$

6.3.1 Training

The Noise2Void (N2V) network is built using CSBDeep Framework [WSB+18], which is followed by a U-Net [RFB15] architecture, batch normalization and activation function. The input and output for the network is a patch of the noisy image. The goal is to predict the center pixel of the image. When the input and the output are the same, the network produces an identical image. Hence, the center pixel of the input patch is masked with a value randomly selected from the neighboring pixel to prevent the network from learning an identity function. This way, the network learns to predict the center pixel value from the *receptive field*. Although this method works, predicting a single pixel for a given image patch is inefficient. Hence, multiple pixels are masked in the input patch, and the network predicts their values. Once the network is trained, it is used on the noisy image to produce a denoised result.

6.3.2 Results

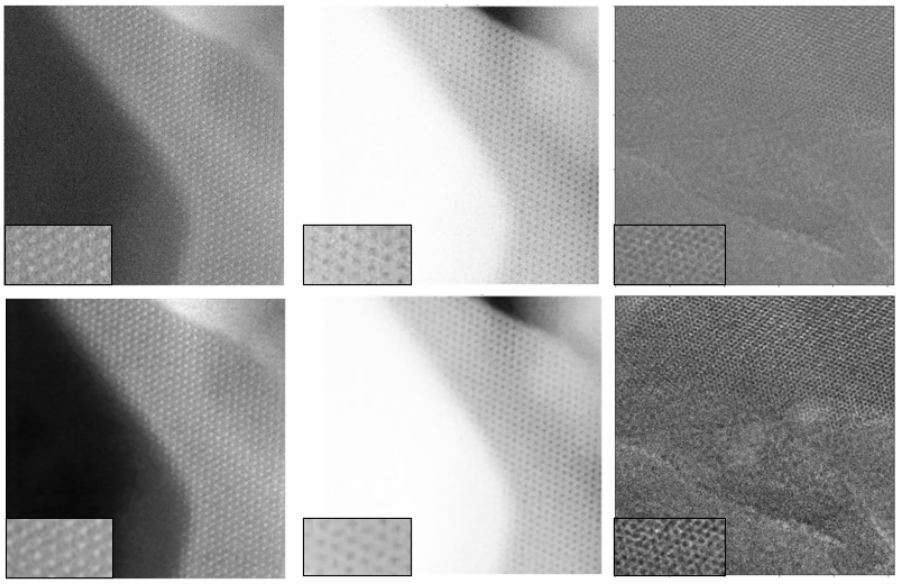

Figure 6.4: Noisy images in the top row and their corresponding denoised images in the bottom row

N2V has gained much popularity lately, especially for the denoising of microscopy images. The main advantage of this method is that it uses single noisy images to perform the denoising task, which is not an option with most other neural network implementations [AC17] [Lef17]. This was important, as the experimental data does not have ground truth data.

N2V has the option to configure the U-Net network. The network was trained with different configurations. The best results were obtained with the network configuration having a kernel size of $3 * 3$, U-Net depth of 3 layers, patch size of $32 * 32$, batch size of 128, and 100 epochs. Mean squared error was used to minimize the cost function, and 6 out of the $32 * 32$ pixels were masked. The results of the denoised images are as shown in figure 6.4.

From the results obtained from applying N2V, it can be observed that the overall quality of the images has improved. In the leftmost image in figure 6.4, the noise on the black region has been suppressed relatively well. In the zoomed-in sections on the bottom left of each image, one can observe that the noise has been suppressed to some extent, and the features like the circular structures (on the leftmost and center images) appear better than their noisy counterpart. However, the shape of these circular structures is not distinctly identifiable, and the features between these circular structures are not recognizable. It is the same with the dark-field image (in the middle). On the rightmost image, the denoised image appears to be still noisy, but the image's contrast has been enhanced, and the overall quality of the image has been improved. The features of the image are more recognizable than its noisier version. On comparing the results obtained from N2V and BM3D, it can be said that BM3D has performed the denoising task slightly better. Since the denoised images are still quite noisy, the results have little use for further scientific analysis.

One of the reasons why the denoised results appear noisy could be that the noise might not be completely pixel-wise independent. Pixel-wise noise independence is one of the assumptions that N2V makes for performing the denoising task. The component of noise, which might not be pixel-wise independent, might still exist in the denoised image. Also, N2V cannot be expected to outperform other denoising methods consistently since it uses

less information for training its network. For instance N2V cannot always outperform the performance of BM3D[KBJ19]. These might be the possible explanations for the results obtained with N2V.

6.4 Summary

The denoising techniques explained in this chapter were an attempt to denoise the experimental images by using existing denoising techniques. A detailed comparison of the results from these methods is done in section 7.4. It can be noted from visual assessment and also from the comparison using Fourier transforms (refer section 7.4) that these denoising methods successfully suppressed the noise and enhanced the features of the image by a small margin. Hence, the results are not significantly better suited for further scientific analysis.

Chapter 7

Non-Local Self-Similarity based image denoising

The denoising results from the previous chapter were only successful in denoising the images by a small margin (refer section 7.4). Hence, the results are often not good enough for further scientific analysis. This was the motivation behind the development and analysis of the new algorithm which is named as "Non-Local Self-Similarity based image denoising".

This algorithm is based on the ideas of Dr.Christoph Pratsch, Helmholtz Zentrum Berlin. It uses self-similarity (similarity between different regions within an image) to average patches that are similar. Averaging patches with similar information suppresses noise that is randomly distributed. Since the noise is assumed to be having zero mean, it cancels out when multiple patches are averaged. However the base signal remains the same throughout and averaging does not disrupt the signal. Hence combining different patches results in denoised pixels close to the actual signal value.

Let x_i be the i^{th} noisy patch, k be the number of patches that are averaged, s_i, and n_i be the signal and noise in the i^{th} patch respectively. Then,

$$x_i = s_i + n_i \tag{7.1}$$

Averaging over k patches results in an expected value,

$$E[\frac{1}{k}\sum_{i}^{k} x_i] = E[\frac{1}{k}\sum_{i}^{k}(s_i + n_i)] \tag{7.2}$$

Since the noise is expected to be having zero mean and the base signal is expected to be the same,

$$E[\frac{1}{k}\sum_{i}^{k} n_i] = 0 \tag{7.3}$$

$$E[\frac{1}{k}\sum_{i}^{k} s_i] = s \tag{7.4}$$

since $s_i = s$ for all i. Ideally this means that,

$$E[\frac{1}{k}\sum_{i}^{k} x_i] = s \tag{7.5}$$

Hence the variance of the averaged patch is small, i.e,

$$Var[\frac{1}{k}\sum_{i}^{k} x_i] = \frac{1}{k^2 - 1}\sum_{i}^{k} Var(n_i) \tag{7.6}$$

since $Var(s_i) = 0$ for all i. Hence averaging patches with the same signal suppresses noise.

This algorithm is inspired by Single Particle Imaging [Bhu17], which is an image processing technique used to analyze low dose Transmission Electron Microscopy (TEM) images of identical but dose sensitive samples. Images of specimens obtained from TEM are often very noisy, and hence it is difficult to interpret the information contained in the image. Averaging several images of the same specimen can remove random noise and make the information more interpretable [Bhu17]. The fact that averaging multiple images with the same information removes noise has been supported in [noa20] too. Similar to Single Particle Imaging, where multiple images with the same specimen information are averaged to enhance the image features, in this algorithm too similar regions within an image are averaged to enhance the image features.

It can also be said that this algorithm is similar to the Non-Local Means [BCM11] algorithm in some aspects. In Non-Local Means, similar patches are found only from the immediate surroundings, but the new " Non-Local Self-Similarity based image denoising" algorithm takes a reference patch and finds similar patches from different regions in the image. The Non-Local Means algorithm is not so non-local, which is it's main drawback. Increasing the search space in Non-Local Means has a massive impact on the computation speed. It can be seen from figure 7.1 that the result obtained from applying Non-Local Means algorithm is still noisy and hence not useful for denoising the experimental images.

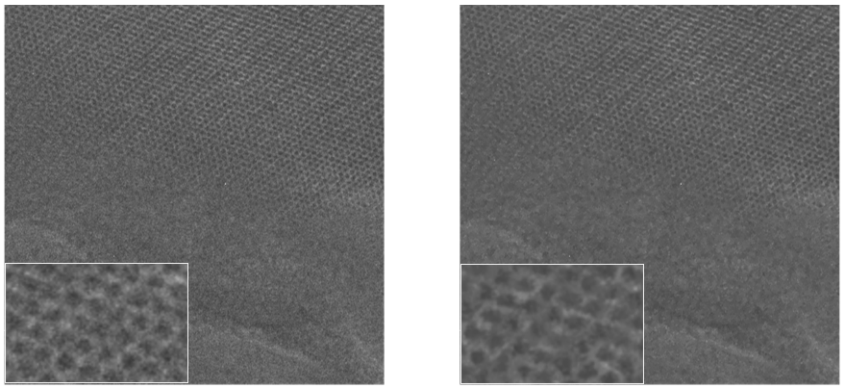

Figure 7.1: Noisy image (left) and it's result (right) obtained from applying Non-Local means

In the following sections, the "Non-Local Self-Similarity based image denoising" algorithm is explained in detail, the results are discussed and analyzed.

7.1 Outline of the algorithm

As described briefly in the above paragraphs, similar patches in a noisy image are identified, and these patches are averaged to get what is called a *reference patch*. Figure 7.2 shows *reference patches* and other patches in an image. The patches that are similar to a *reference patch* are connected by a

line. In other words, a *reference patch* represents the patches connected to it, and the patches connected to a same *reference patch* form a group.

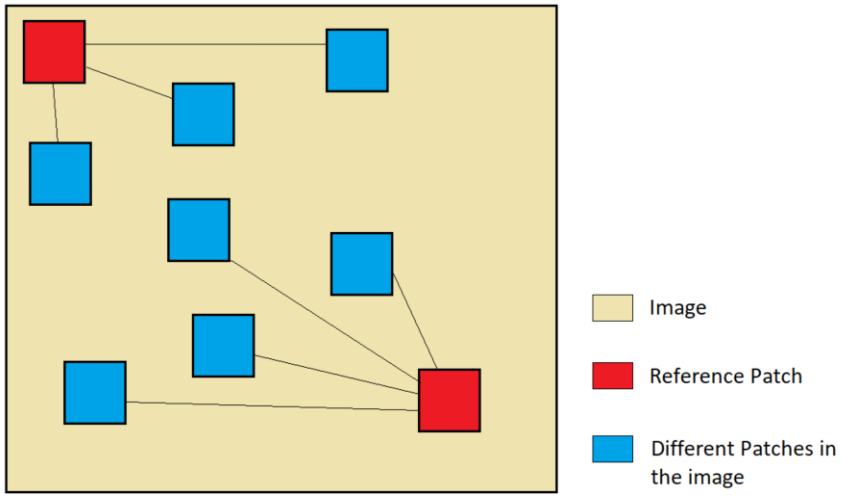

Figure 7.2: The blue patches linked together represent similar regions in an image (identified using the template matching algorithm). These similar patches linked together are averaged to get a *reference patch*

The Non-Local Self-Similarity based image denoising algorithm mainly uses template matching and clustering to perform the denoising task. The flowchart of the algorithm is shown in figure 7.3.

The algorithm begins with the initialization part. Input for the algorithm can be a single image or a stack of images. Once the images are loaded, the initial *reference patches* are obtained from the images. A *reference patch* in this algorithm can be thought of as a template with size $m * m$ as shown in figure 7.2. These *reference patches* are used for the template matching algorithm to get other patches similar to them, which is explained later.

For example, initial *reference patches* can be as shown in figure 7.4. From the input images, initial *reference patches* are generated from random positions. The initial *reference patches* ideally should represent different reoccurring regions of an image. Hence, even though the initialization is random, it is ensured that the initial templates are spread across the image.

The user can choose the patch size, but a patch size of $46 * 46$ was used for most of our experiments. The reason for this choice is explained in section 7.2.

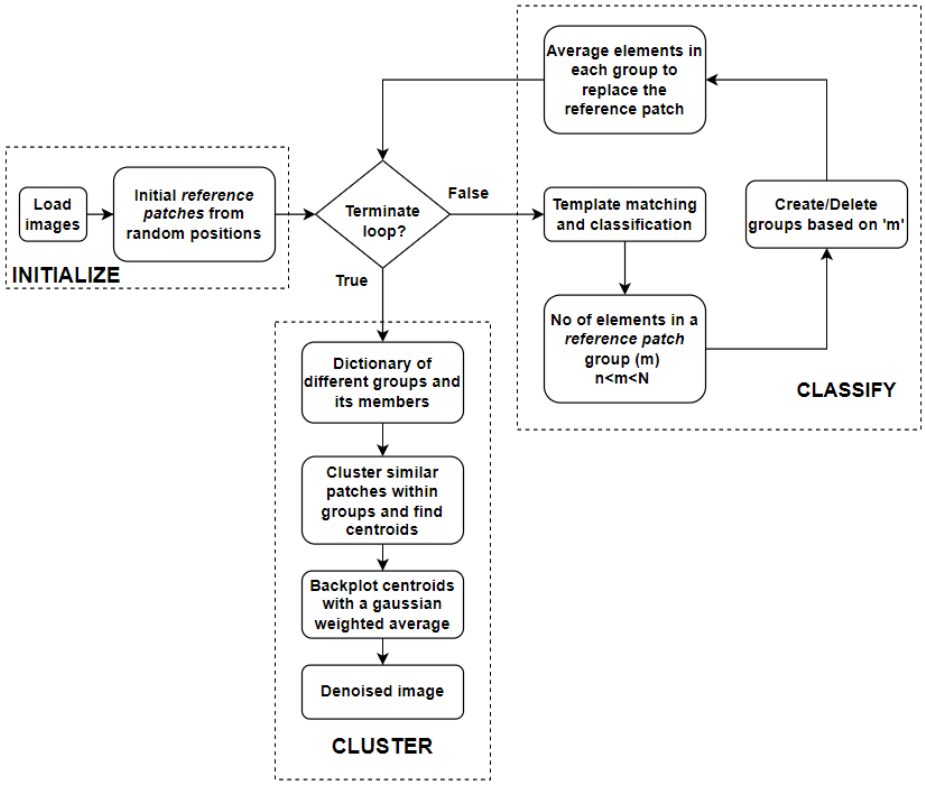

Figure 7.3: Flowchart of the algorithm

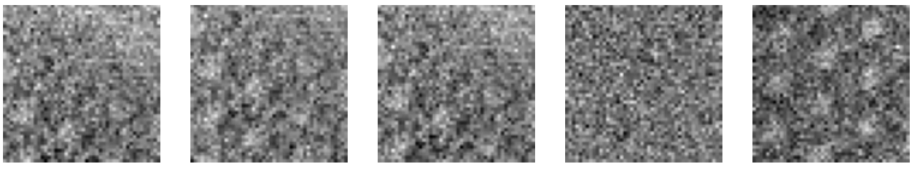

Figure 7.4: Initial *reference patches* (of size 46*46 pixels), taken from different regions of the input image

The denoising algorithm uses the template-matching from the '*scikit-image*' python library to match *reference patches* with the image. This template-matching algorithm is an implementation from [Lew01], which is based on fast normalized cross correlation between two images. This method is a fast way of matching features by transforming the data into the frequency domain by the application of Fast Fourier Transform (FFT). Figure 7.5 demonstrates the working of the template-matching algorithm. An image, a template taken from the image (i.e., *reference patch* in this algorithm), and the corresponding result have been shown. The maximum value in the result which is marked in red, represents the template's location in the image. It corresponds to the top-left corner of the template. Values close to the maximum represent the patches similar to the template. The following paragraph explains how different patches in the images are classified.

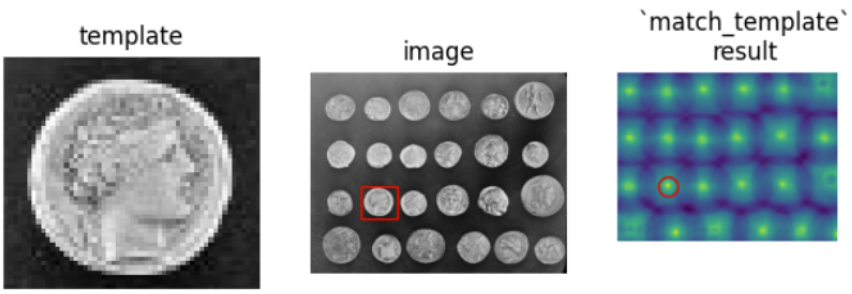

Figure 7.5: Template matching example [2]

Initialization process is followed by the "classify" section shown in figure 7.3. Here, patches at every position in the image are classified into different groups, and these groups are represented by *reference patches*. For classification, the result of the template-matching [Lew01] algorithm is used. If x and y represent a pixel location, then the patch at (x, y) is obtained from pixel locations (x to $x+m$, y to $y+m$), where $m*m$ is the patch size. When the template matching algorithm runs, each of such $m * m$ patches in the image is compared with every $m * m$ *reference patch*. For an image of size

[2] https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_template.html

$p * q$ and a *reference patch* of size $m * m$, the template matching result has a size $(p - m + 1) * (q - m + 1)$. The result has values between -1 and 1 as the results are normalized. The location with a perfect match is represented by value 1. The patches similar to the *reference patch* are represented by values close to 1.

The "classify" section of figure 7.3 begins with k images of size $p*q$, that have to be denoised and n initial *reference patches* of size $m * m$. For every k set of images, template matching is performed with each of the n *reference patch*. Hence for every image, n results of shape $(p-m+1)*(q-m+1)$ are obtained. These results are stacked into an $n*(p-m+1)*(q-m+1)$ array. From this array, the *reference patch* that matches the best at every location is identified. Hence, the $n * (p - m + 1) * (q - m + 1)$ array is reduced to a $(p-m+1)*(q-m+1)$ array which contains the best fitting reference path index at every position. Now, patches in the image similar to the *reference patches* are identified. The patches represented by the same *reference patch* form a group as shown in figure 7.2. This process is repeated for all the k images.

Although patches at every location are classified to belong into a group, all these patches are not considered. The patches which have a strong correlation with one of the *reference patches* are prioritized. This is done by selecting patches that are close to the maxima in the template matching result. When these patches with strong correlation are selected, the patches around them in the close neighborhood are deleted. That is, if a patch is selected from (x, y), then no patches in the region from $(x - a$ to $x + a, y - a$ to $y + a)$ are selected, where $(a \approx m/4)$ with $m*m$ being the patch size. This gap between patches is used to remove redundant information. Removing redundant information improves the computation speed without affecting the denoising quality.

Once groups of similar patches are formed, the groups which have very few patches are deleted. The reason for this deletion is that there will not be a great deal of denoising if very few patches are averaged. Additionally groups with few patches can slow down the algorithm without significant improvement in the denoising quality. On the other hand, if there are too

many patches in a group, then such groups are split into multiple groups. This splitting is done to avoid generalizing too many patches as one. Averaging over a large group, even with members having minor differences in the signal, can lead to the disruption of the signal. Limiting the patches in a group also helps in the clustering performance, which is discussed later. Each of the old *reference patches* is now replaced by the average of all the members in that group. In the next iteration, template matching and classification steps repeat with these new *reference patches*. One can observe that every iteration may generate new *reference patches*. New *reference patches* are generated when big groups are split into smaller groups. The "classify" section terminates when the total number of *reference patches* in three consecutive iterations remains the same or when the program completes 15 iterations. It has been observed that running the loop 15 times is enough to generate *reference patches* representing different regions of an image. Figure 7.6 shows the *reference patches* generated after 15 iterations. On comparing figure 7.4 and figure 7.6, one can observe that the noise in the final *reference patches* has been suppressed.

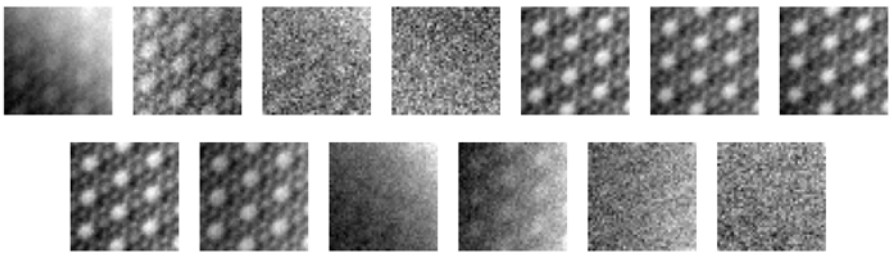

Figure 7.6: Final *reference patches*

After running the "classify" section multiple times, the final *reference patches* are used to replace the patches in their corresponding groups. The denoised image is constructed by using these final *reference patches*. Since *reference patches* can overlap during reconstruction, a $2D$ Gaussian weight is used. The information at the center of a patch is important than the edges. While back plotting, if multiple patches overlap, a weighted average is done. Weighted average with Gaussian weights ensures that the back plotted

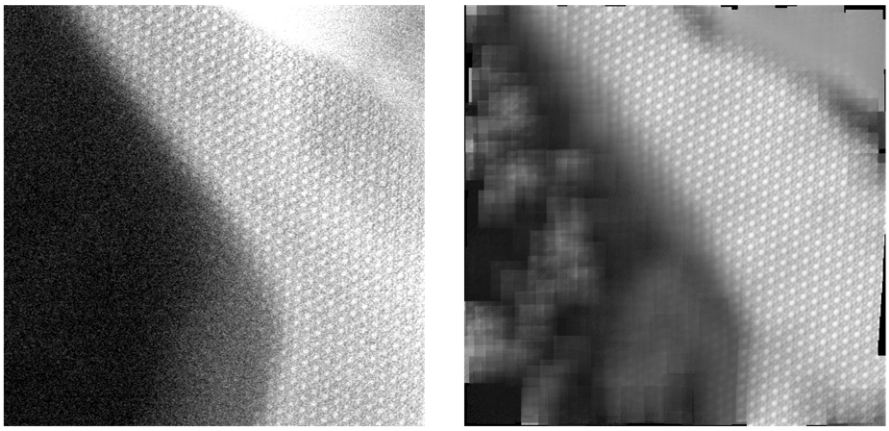

Figure 7.7: Noisy input and it's corresponding back plot obtained from back plotting the *reference templates* (before the clustering step)

images do not have sharp edges from the patches used, as the information from the patch edges is suppressed. The back plotted image can be seen in figure 7.7. The following can be inferred from the back plot:

- A few final *reference patches* (shown in figure 7.6) cannot be used to reconstruct the whole image. There can be patches in a group that are less similar to the *reference patch*. When these dissimilar patches are averaged, the average might differ from the actual patch by a large extent.
- Averaging a large range of data also changes the range of pixel values, altering the brightness after reconstruction.
- The edges contain some black spots. These are the regions that are not linked to any *reference patch*.

The first two problems can be solved by averaging over a small group with very closely matched patches. Splitting large groups into smaller ones is done by applying clustering within a large group.

Hierarchical clustering[3] [Sha19] is used to cluster the large group into

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.fcluster.html

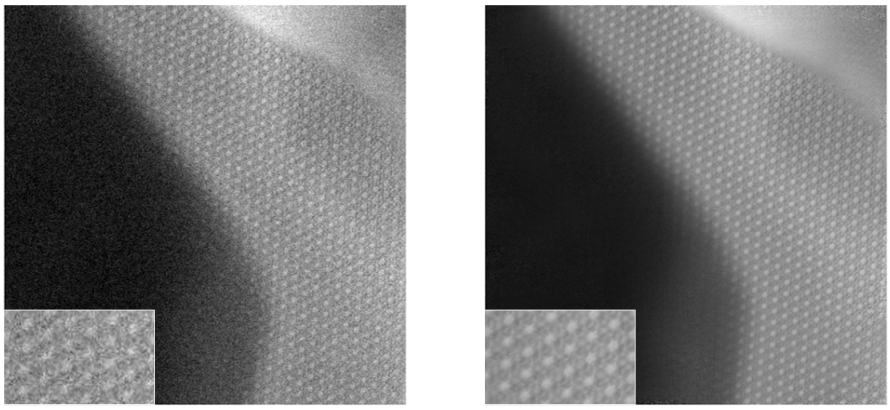

Figure 7.8: Noisy input and it's corresponding denoised result

smaller ones. Hierarchical clustering performs clustering by finding the Euclidean distance between each sample member. A tree-like dendrogram is built from the distances, and then clustering is done based on the number of clusters desired. Since distance is used to separate samples, clusters with unequal sizes can also be formed, unlike the k-means clustering [Sha19], which is also a popular clustering method. Having clusters with unequal sizes is essential as some groups (of patches) may contain very few dissimilar patches. The following paragraph explains how clustering is applied in the denoising algorithm.

The problems seen in figure 7.7 are addressed in the "cluster" section seen in figure 7.3. Clustering is applied within every group (represented by the final *reference patch*) to create smaller subgroups, i.e., clusters are created within large groups. The number of clusters in a group is determined by a constant 'c' that the user can set. From 'c', the number of clusters in a group is obtained by dividing the number of patches in a group by 'c^2'. That is, the signal-to-noise ratio can be adjusted by changing the 'c' value (see also section 7.2). Hierarchical clustering is done, and the corresponding centroids are found for every cluster. Few centroids now represent a large group that was before represented by a single *reference patch*. The centroids of different groups are used to reconstruct the image, as they replace the patches in their cluster. Centroids used for back plotting are multiplied by a $2D$ Gaussian

weight. A weighted average is performed when more than one centroid is used at a position (as explained before). Gaussian weight smoothens the edges of the centroids, thus preventing artifacts in the reconstructed image. Clustering solves the first two of the three problems that were caused by using only template matching.

The regions without a *reference patch* can be seen as black regions around the edges in figure 7.7. This problem is rectified by simply copying the pixel values from the noisy input. Finally, a denoised image is obtained, which is shown in figure 7.8.

The denoising algorithm can be seen in appendix A under algorithm 1. The Python code for the denoising algorithm can be found on github[4].

7.2 Parameters of the algorithm and stability

The algorithm uses a few values or constants (the algorithm's parameters) that the user can define. Following is a description of the effect these parameters have on the algorithm:

- **Size of the patches**: Patches represent features in an image. If the patch size is too small, the features might not be well represented in a patch, resulting in template matching producing incorrect results. On the other hand, a patch size that is too large will result in having multiple features in a patch which again will cause problems during template matching. In either case, averaging incorrect patch members in the group will not suppress the noise.

  A patch size of $46 * 46$ performed the template matching task reasonably well, and hence this was the size used for most of the experiments. However, different patch sizes were used to test how the result varies with a change in patch size. It was observed that the algorithm produced good results for patch sizes from $30 * 30$ to $60 * 60$. Changing the patch size affects the computation speed too. The algorithm is faster when a larger patch size is used. The enhancement in the speed

[4] https://github.com/mbanil/img-denoiser

can be attributed to the enhancement in the computation speed of the template matching algorithm. As the patch size increases, the search space reduces, and hence computation speed of the template matching algorithm is faster [Lew01]. The variation of computation time with changes in the patch size can be seen in figure 7.9.

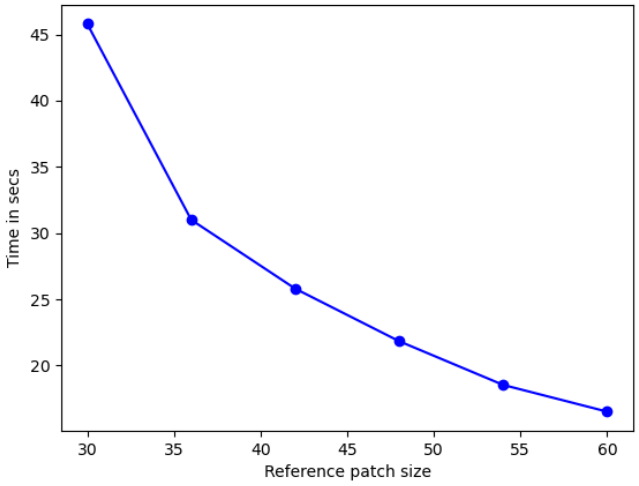

Figure 7.9: Variation of computation time with changes in the patch size

- **Position of the initial patches**: The position of the initial patches has to be selected such that the patches represent different regions in the image. These patches are used as initial *reference patches* to represent a group. The patches within a group should be similar, but those from a different group should be dissimilar. Hence it is crucial to choose the initial patches that differ from each other. These dissimilar patches are selected by randomly initializing the templates such that no two templates are close to each other.
- **Loop counter for running the "classify" section (figure 7.3)**: As mentioned earlier, the loop is made to run until the number of *reference patches* generated stays the same for three consecutive iterations. This can be considered as the termination condition for the

loop. However, the algorithm also produced satisfactory results even when the loop was made to run for five iterations. The termination condition used generates maximum *reference patches* with minimum time complexity. If there are more *reference patches* generated, then the distinct regions of the image are well represented resulting in minimal artifacts in the denoised result.

- **Minimum number of patches in a group**: The minimum number of patches in a group (represented by a *reference patch*) is set to four. As mentioned earlier, if a group has less than four patches, such groups are deleted. When there are very few patches, averaging them will not significantly suppress the noise.

- **Maximum number of patches in a group**: The maximum number of patches in a group (represented by a *reference patch*) is set to 100. There are two reasons to limit this number. The first reason is to avoid overgeneralization. If the group size is big, a single *reference patch* will represent too many patches, which is undesirable. The second reason is that too many patches in a group affect the computation speed of clustering. Hence the maximum number of patches in a group is limited. This number also is a criterion for generating new groups and thus new *reference patches* (obtained by averaging patches in new groups). If a group contains patches more than this number, the group is split into multiple groups.

  In the case of image stacks, this parameter is increased according to the number of images in a stack. This increment is done to improve the overall speed of the algorithm. More groups are generated if this parameter has a lower value, resulting in more *reference patches*. The number of *reference patches* is proportional to the number of times the template matching algorithm is called. Too many *reference patches* can affect the overall computation speed of the algorithm. Hence a trade-off is required.

- **The number of clusters within a group**: Once the groups are

formed after classification, clustering is done further to divide a big group into multiple smaller clusters. The number of clusters within a group (represented by a *reference patch*) is a crucial parameter for the denoising task. This parameter must be set based on how much self-similarity is present in the input image. If there is much self-similarity, few clusters can be made because more patches are very close to each other. Hence a centroid can be used to represent more patches. If there is not much self-similarity, having few clusters can make dissimilar patches fall into the same cluster. When a centroid represents dissimilar patches, the output might look patchy, which can be seen in figure 7.10. Hence having very few clusters is similar to the results obtained after classification, i.e., figure 7.7. On the other hand, if too many clusters are created, denoising does not work as there are very few patches that generate the centroids. When there are many clusters, the output still contains a lot of noise, as can be seen in figure 7.11. Hence it is crucial to choose the number of clusters correctly.

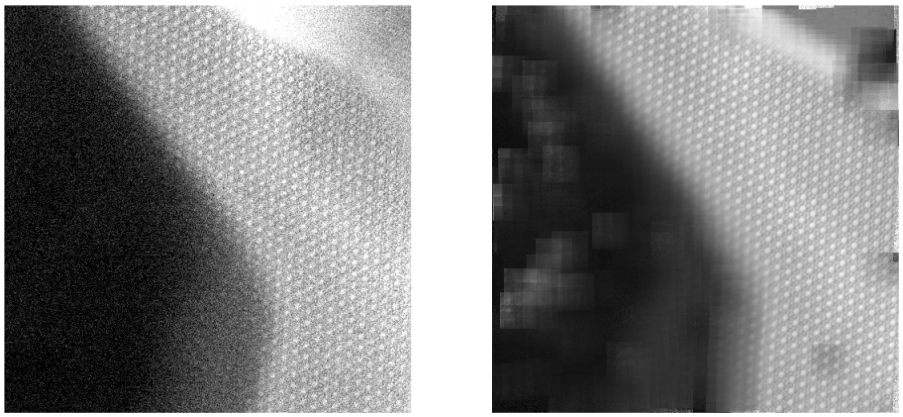

Figure 7.10: Noisy input and the result when very few clusters were created within a group

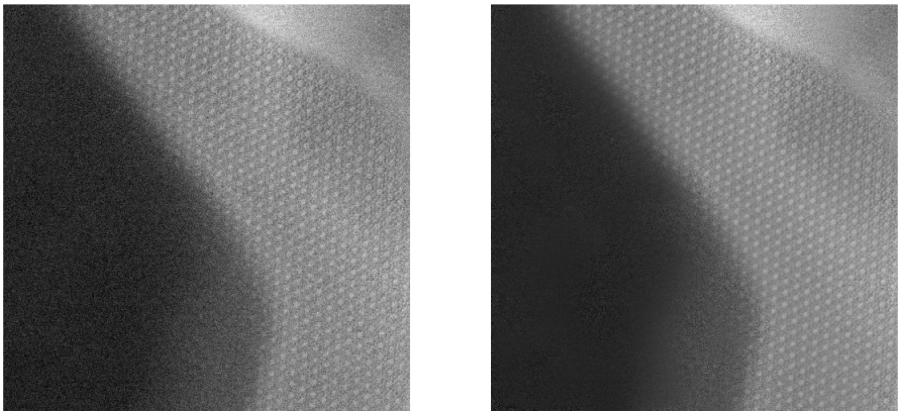

Figure 7.11: Noisy input and the result when very many clusters were created within a group

The number of clusters is calculated by dividing the group member's count by the square of this parameter. If this parameter has a higher value, fewer clusters are created, and hence the signal-to-noise ratio is high. Similarly, the signal-to-noise ratio decreases if this parameter is small.

It should also be noted that the same result was always produced upon running the algorithm multiple times with the same parameters and the same input. This means that the algorithm does not introduce any randomness in between.

7.3 Analysis of the results

Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Metric (SSIM) are popularly used to measure the quality of image denoising. Both of these techniques required an original noise-free image to calculate the quality of denoising. Since an original noise-free image is not present for the experimental data used, Fourier image analysis and visual assessment are used to examine the quality of denoising. FFT[5] was applied on the denoised

[5] https://numpy.org/doc/stable/reference/generated/numpy.fft.fft2.html

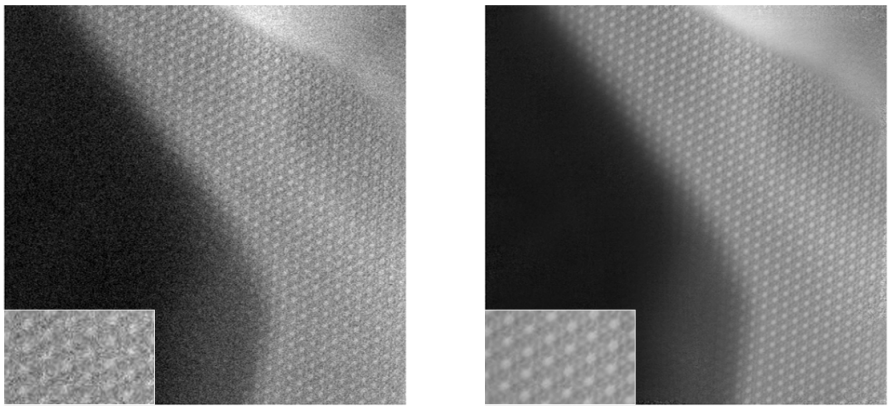

Figure 7.12: Noisy input and it's corresponding denoised result

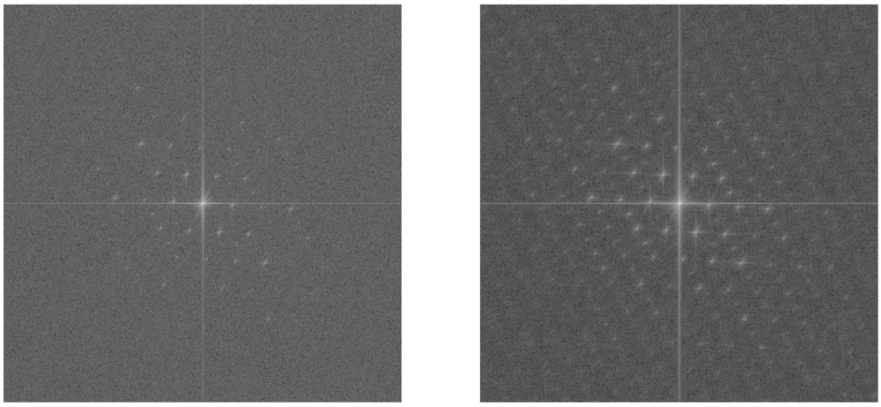

Figure 7.13: FFT of the noisy input and it's corresponding denoised result

results to obtain the Fourier transform.

An example of the input image and its denoised result are as shown in figure 7.12. The FFT of these images are as shown in figure 7.13. The vertical and the horizontal lines passing through the center of the FFT images is because of the square images used [Ste97]. The white structures in the FFT of the noisy image (figure 7.13) represent the features. It can be seen that these white structures have been enhanced in the FFT of the denoised image (figure 7.13). In the noisy FFT, the white structures are visible only at the center. As the observer moves away from the center, the image looks

mostly blank, and there are no patterns visible. However, in the denoised FFT, the white structures present close to the center appear much bigger and brighter, and as the observer moves away from the center, the denoised FFT still shows some white structures, some of which are completely absent in the noisy FFT. This enhancement of white structures means that the denoising algorithm has extracted features from the noisy data, thus suppressing noise. This is also supported by the figure 7.12. In figure 7.12, on observing the noisy image, the shape of the circular structures is hardly recognizable. However, in the denoised image, the shape of these circular structures can be distinctly recognized. The features between the circular structures have also been reconstructed fairly well. These features are not recognizable in the noisy input. Additionally, the noise seen in the plain black region (towards the left of the image) has also been successfully suppressed. Hence, visual assessment of the results supports the explanations one can draw from observing the FFT of the images.

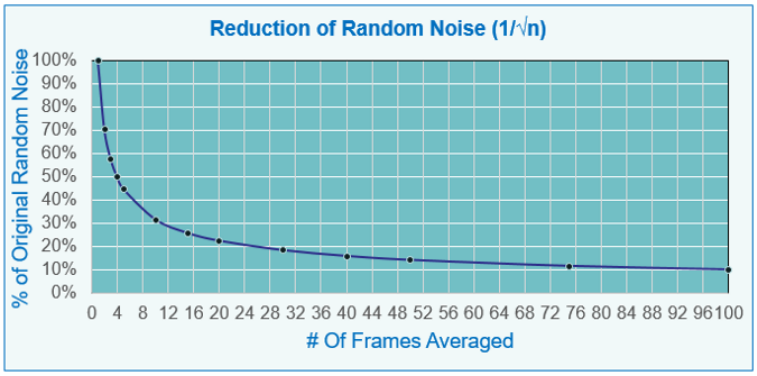

Figure 7.14: Suppression of random noise v/s the number of frames averaged [noa20]

The main reason for noise suppression is the averaging of very similar patches in the image. Suppression of noise is proportional to $\sqrt{n}$, where n is the number of patches used [BCM11] [noa20]. This result is visualized in the graph shown in 7.14. The percentage random noise reduction with the number of frames used to find the average can be seen. Further analysis of

the denoising process is explained in the following.

After the "classify" part of the algorithm, patches at different positions of an image are grouped based on the *reference patches*. The different colored squares in figure 7.15 represent different groups. Hence different squares of the same color spatially represent the different patches that are represented by the same *reference patch*. It should be noted that the squares represent the top-left position of the patch. Hence the right and bottom borders are empty. It can be observed that many groups have many members. Hence averaging the members of each group will certainly reduce the noise (as supported by the graph in figure 7.14). However, using a *reference patch* that represents a wide range of patches can introduce artifacts. It also varies the brightness and contrast of the image. These problems can be seen in figure 7.7. Hence clustering is done to make smaller subgroups.

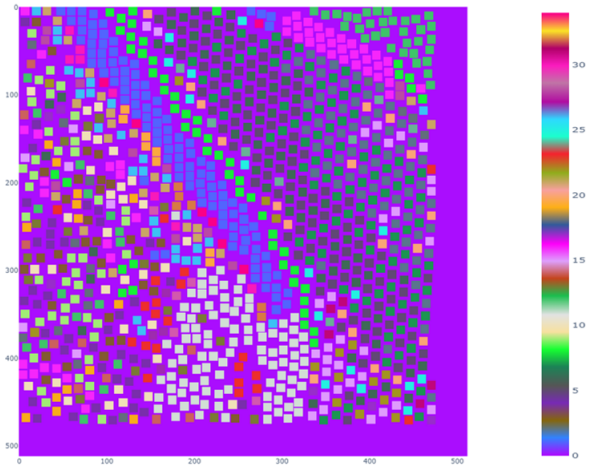

Figure 7.15: Visualization of the similar patches in the image. Same colored squares represent similar patches (forming a group) and hence different colors of the squares represent different groups.

Consider a group of patches that are represented by the same colored squares in figure 7.15. The group members become the samples for clustering, and the number of clusters depends on the parameter 'c' that the user can define. There are around 30 groups shown, and hence clustering is done

around 30 times. As explained earlier, hierarchical clustering is the clustering method used. The parameter 'c' should be chosen based on the extent of self-similarity in the input image. For this particular image shown in figure 7.12, a value of 2.7 was used. Upon experimenting, the ideal range for this parameter was between 1.5 (for images with less similar components) and 2.7 (for images with more similar components). The number of clusters in a group is found by,

$$w = \left\lceil \frac{n}{c^2} \right\rceil \tag{7.7}$$

where w is the number of clusters, n is the number of group members, and $\lceil\rceil$ represents the conversion to the next highest integer.

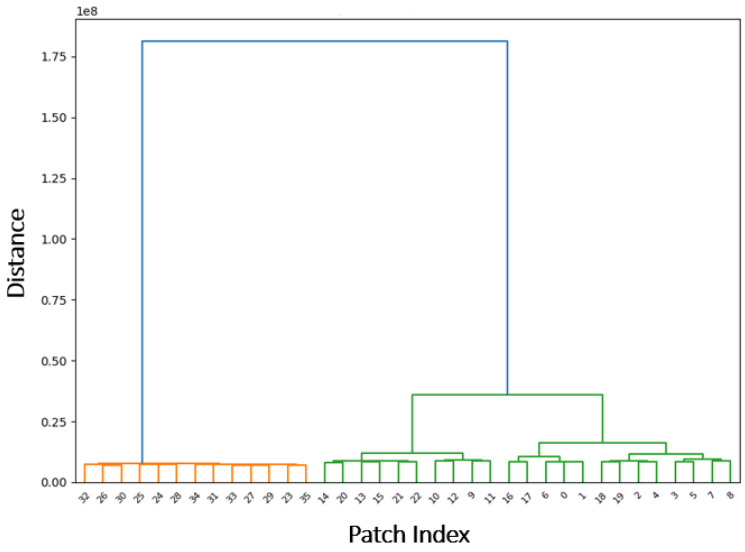

Figure 7.16: Dendrogram obtained from hierarchical clustering on a group. The group is split into multiple clusters.

The figure 7.16 shows the result of applying hierarchical clustering on a group. The X-axis represents the group's different image patches, and the Y-axis represents the distance. It can be observed that the group consists of two distinct clusters, one marked in orange and the other marked in green.

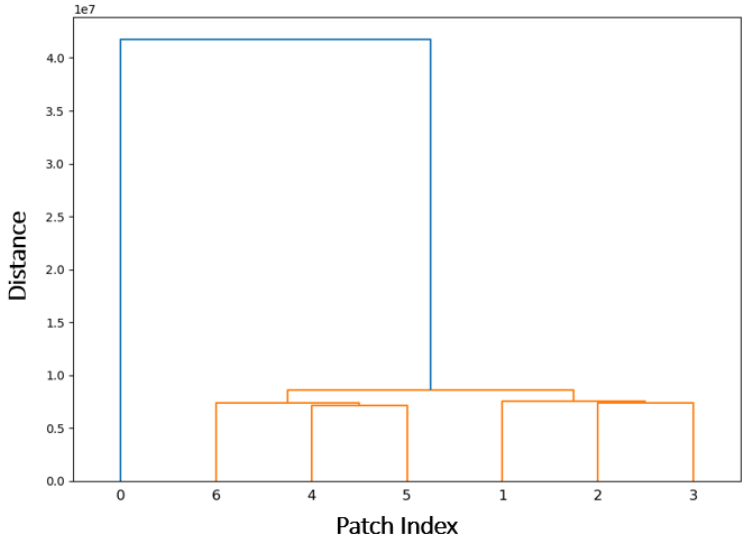

Figure 7.17: Dendrogram obtained from hierarchical clustering on a group. The group is not split and hence a single cluster is obtained.

Having the same *reference patch* for the entire group can introduce artifacts. On performing clustering, we get at most five clusters (from equation 7.7 with 36 members and 2.7 as the clustering parameter). At most, five clusters mean that different centroids now represent similar patches.

However, sometimes clustering can also fail to create smaller clusters when the clustering parameter is not chosen correctly. In figure 7.17, there are seven members, and choosing a clustering parameter of 2.7 will result in one cluster, which is not desirable as one of the members is dissimilar from the rest of the group. Clustering parameter having a higher value has an advantage as the noise suppression is more. However, it can introduce some artifacts if there is not much self-similarity in the image. Hence there is always a trade-off.

Once the different centroids are obtained, they are back plotted to get the denoised image. Centroids overlap while back plotting and are averaged with a $2D$ Gaussian weight. Averaging here further suppresses the noise as per the plot shown in figure 7.14. The number of patches that are averaged at every

pixel location is visualized in figure 7.18. The visualization also considers the number of patches that form a centroid in the clustering process. One can observe that over 200 patches have been averaged at certain regions, which suppresses the noise by over 90% according to the statistic shown in figure 7.14.

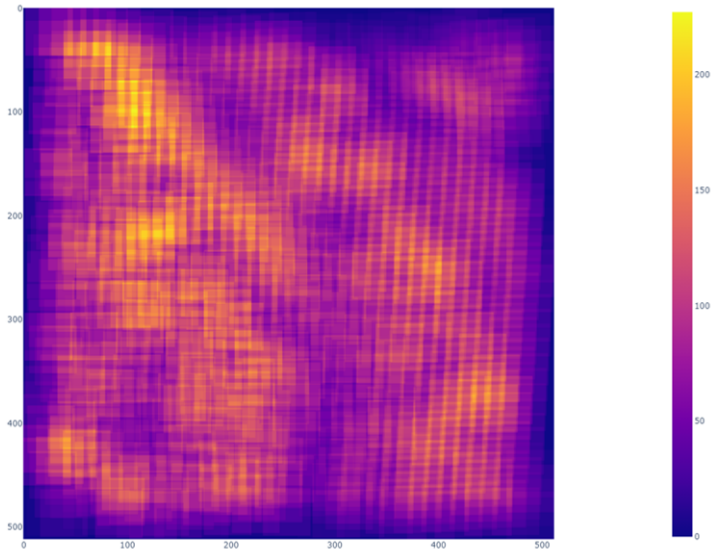

Figure 7.18: Each location in the visualization represents the total number of pixels that are averaged at every location to get the denoised result.

From the above discussions, it can be observed that the algorithm can introduce artifacts. It is important to recognize this to determine if the denoised image is satisfactory. Since it is challenging to detect minute artifacts from Fourier analysis, a confidence map was developed. This confidence map is based on the variance within each cluster obtained after applying hierarchical clustering. The variance should be small if the centroid is a good representation of its member patches. Also, a good centroid should not have any patterns in its variance. Patterns in the variance show that the centroid does not generalize its members well. Figure 7.19 shows the centroids which is a good generalization (centroid-1) and which is not (centroid-2). The centroid-2 has been taken for demonstration purposes by having very

few clusters in a group.

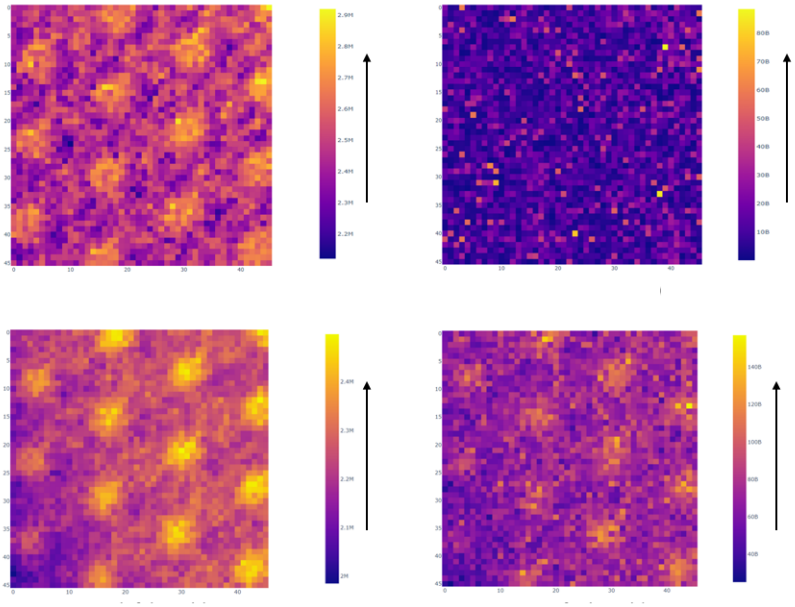

Figure 7.19: Centroid_1 and variance_1 in the top row - Variance is low in this case which is desirable. Centroid_2 and variance_2 in the bottom row - Variance is high and has structures in this case which is undesirable.

The confidence map of the denoised image is calculated by combining the variances for different centroids as they are back plotted. The variances are combined by using the method proposed in [CGL82]. Using [CGL82], variances can be updated when a new sample is added. The algorithm for combining the variances has been mentioned in appendix A under algorithm 2.

The overall variance map, i.e., the confidence map of a denoised image is as shown in figure 7.20. This result has been produced for demonstration purpose with a very few initial *reference patches* selected very close to each other. The bright regions in the confidence map correspond to the denoised image artifacts. For instance, a bright region can be seen in the confidence map at the top-right position. An artifact can be found by inspecting the same region on the denoised image. Similarly, irregularities in the black

region on the left side of the denoised image can be recognized by the brighter regions of the confidence map.

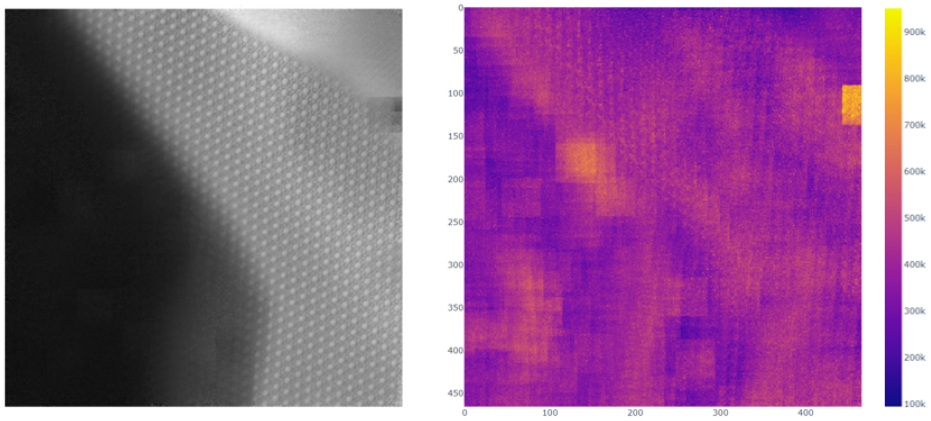

Figure 7.20: Denoised image and it's confidence map

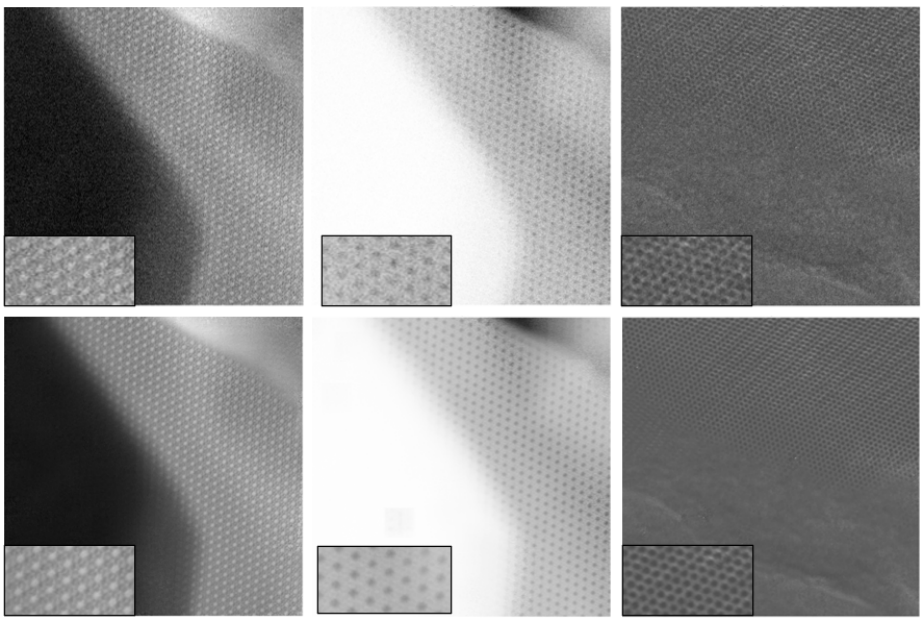

Figure 7.21: Noisy images in the top row and their corresponding denoised images in the bottom row

The results of the algorithm for all the experimental images are as shown in figure 7.21. On inspection, it can be observed that the denoising task has been carried out fairly well, suppressing the noise and enhancing the image features.

Denoised results of other microscopy images can be seen in Appendix B.

7.4 Comparison of results

The noisy image, the results from all the methods and their Fourier transforms can be seen in figure 7.22. The result from the generic autoencoder looks a bit noisy even though the circular structures are prominently visible than it's noisier counterpart. The substructures between the circular structures are not visible. The black region also looks noisy. The Fourier transform supports this argument as there isn't any remarkable enhancement in the white structures at the center. The edges appear bright which can be attributed to the noise still existing or additionally added by the denoising process.

N2V does a good job in reconstructing the circular structures and removing the noise in the black region. However, the substructures have not been reconstructed well. The Fourier transform shows that the enhancement of white structures at the center. The edges mostly look dark thus representing the suppression of noise.

BM3D does a slightly better job than N2V. This is also supported by the Fourier transform. However the substructures are not reconstructed well.

Non-Local Self-Similarity based denoising has produced the best result among the methods that have been tested on the experimental data. This method does a good job in reconstructing most of the structures from the noisy image. The Fourier transform also shows enhancement in the white structures which is the most compared to those from the other methods. Upon observing the result in figure 7.12, the substructures between the circular structures can also be seen.

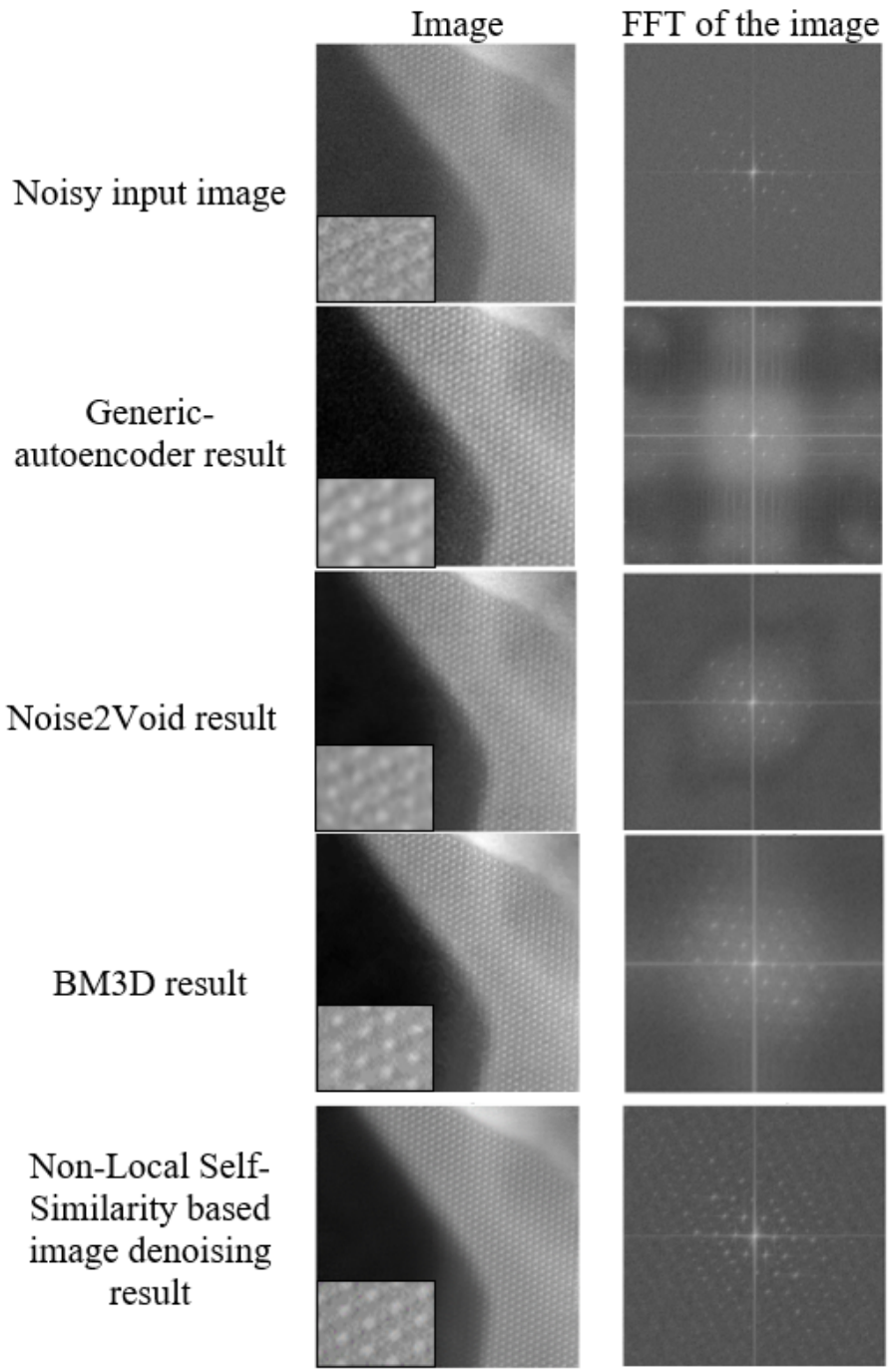

Figure 7.22: Comparison of results obtained from different methods

7.5 Results on natural images

On observing the successful results from applying the Non-Local Self Similarity based denoising algorithm on electron microscopy images, curious readers would be interested in the results from applying the algorithm on normal images. The figure 7.23 shows a normal image and its denoised result. The image is monochrome with pixel values ranging from 0 to 255.

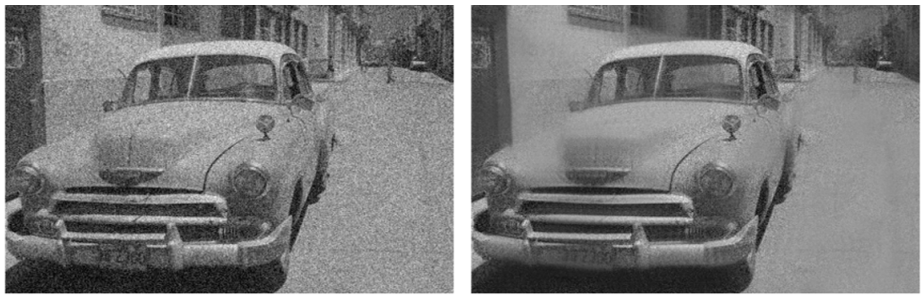

Figure 7.23: Noisy image (left) and its denoised result (right).

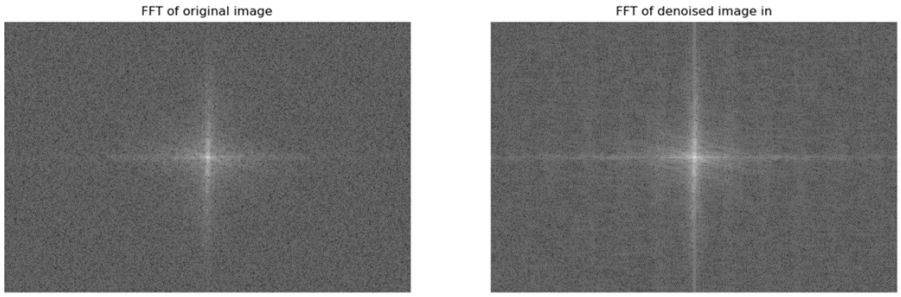

Figure 7.24: FFT of the noisy image (left) and FFT its denoised result (right).

The denoising algorithm has done a fairly good job in denoising the image. It should be noted that the main criterion for the denoising algorithm to work successfully is the presence of self-similarity in images. This is the reason why the noise has been suppressed fairly well in the regions like the hood of the car, grill of the car, road beside the car, etc. The algorithm tries to find similar patches and averages them to suppress noise. If there

are many similar patches, averaging takes place with lots of patches and hence noise suppression is more. Similarly, if there are less similar patches, the noise suppression is less. This is the reason why noise suppression is different at different regions of the image. For instance the side windows of the car or the region at the end of the road have less similar regions. Hence there is not a great deal of denoising in these regions. However, it is important to note that the algorithm tries to suppress the noise wherever possible and retains the regions which do not have much similarity in the image. The algorithm also does a good job in retaining the overall structure of the image.

Figure 7.24 shows the FFTs of the noisy image and its denoised result. The structural enhancement can be verified by comparing the FFTs in figure 7.24. The white structures in the denoised FFT are brighter. The vertical white structure on either side of the central vertical line which are absent in the noisy FFT can be observed in the denoised FFT. One can conclude that this algorithm can be used on normal images, but the best results are obtained if there is self-similarity in images.

7.5.1 Comparison of the results using natural images

In order to compare the denoising methods used so far in a quantitative way, an image represented by 8 bit data format was chosen. The image has repeating structures, which is similar to the experimental images. A noisy version of the image was generated by adding Gaussian noise with mean equals to 0 and standard deviation equals to 0.2. This noisy image was used on all the denoising methods and the denoised results obtained were compared with the original clean image by using Peak Signal-to-Noise Ratio (PSNR). Using PSNR, a numeric value can be obtained from which the method that performs the best can be easily identified.

The clean image, it's noisier version and the comparison of the results obtained from different methods can be seen in figure 7.25. From figure 7.25, one can observe that the auto-encoder was able to denoise the image only by a small margin. N2V and BM3D do a better job at denoising the image. The structures in these denoised images appear better than those in

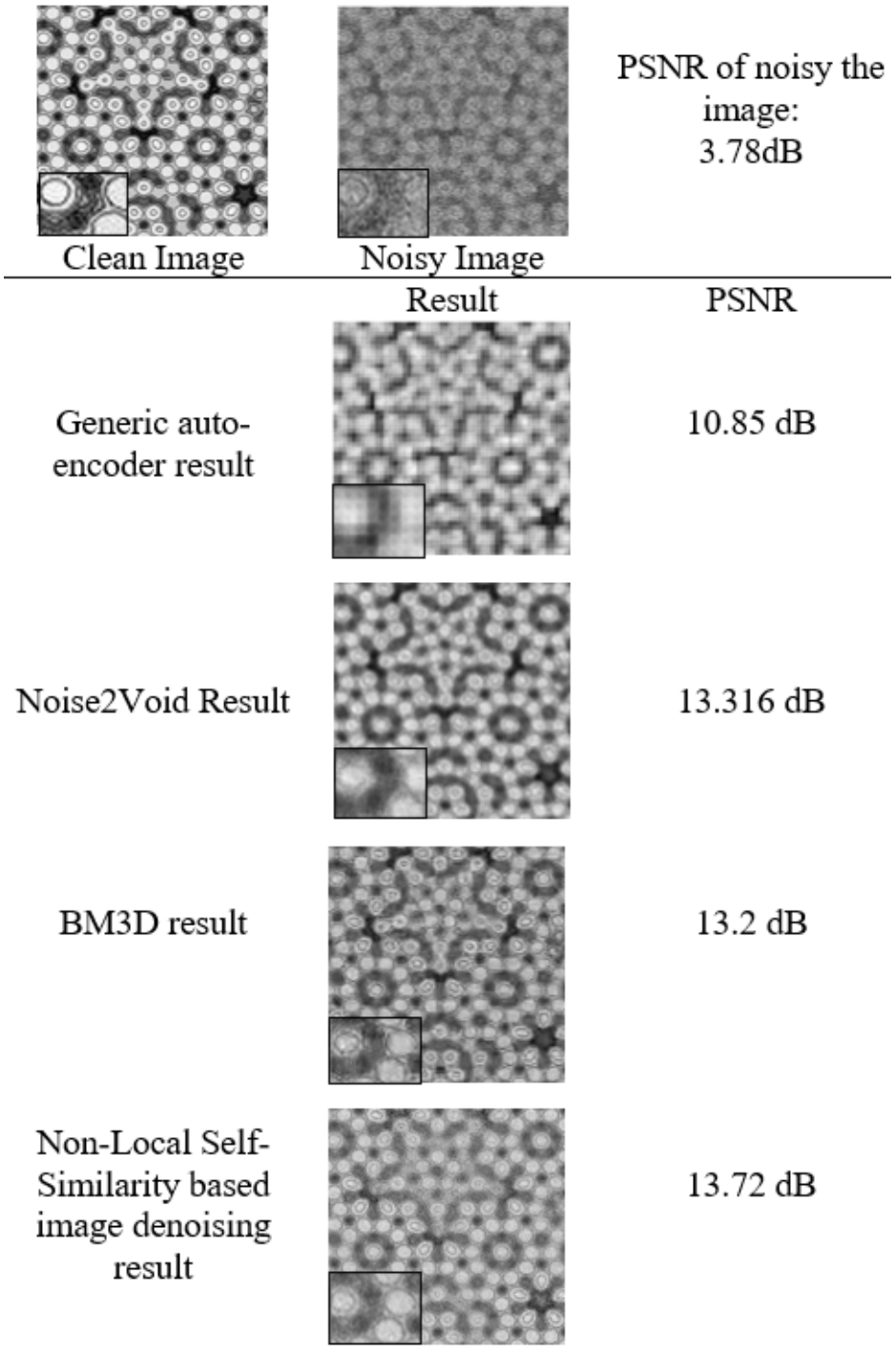

Figure 7.25: Comparison of natural image results

the auto-encoder's result. However Non-local self similarity based denoising algorithm does the best job compared to the other methods. The structures are more clear compared to the results obtained from other methods. The PSNR value supports the argument as it is the highest for the Non-local self similarity based denoising method. The PSNR values of other methods can also be seen in figure 7.25.

7.6 Run-time performance of the algorithm

The algorithm's performance can be evaluated based on the time or the memory required to run the algorithm entirely. The algorithm's performance is as vital as the quality of denoising. A user would not want to run the algorithm for a long time to obtain a denoised image. The algorithm's performance concerning time was evaluated as it is of more importance to the user. The time complexity of the algorithm determines the algorithm's performance concerning time.

According to Wikipedia[6], time complexity is defined as follows.

"In computer science, the time complexity is the computational complexity that describes the amount of computer time it takes to run an algorithm. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, supposing that each elementary operation takes a fixed amount of time to perform. Thus, the amount of time taken and the number of elementary operations performed by the algorithm are taken to be related by a constant factor".

The algorithm's running time varies according to the input size. To calculate the time complexity of an algorithm, the computation time for various input sizes was measured. Since the computation time is not consistent for small inputs, computation time is generally calculated for large input sizes. A graph of computation time versus input is plotted to determine the time complexity. Usually, the algorithm is made to run multiple times with each input size, and the computation time is averaged, or the worst computation time is considered [Sip13].

[6] https://en.wikipedia.org/wiki/Time_complexity

Time complexity is usually represented using big O notation. For example $O(n)$, $O(n^2)$, $O(n \log n)$ represent linear, quadratic and logarithmic time algorithms respectively, where n represents the number of elements in the input.

7.6.1 Time Complexity of the algorithm

A plot of computation time versus input size was plotted to understand the denoising algorithm's time complexity. The algorithm was made to run three times with the same input image. The time taken for the algorithm to run completely was averaged with the input image sizes varying from $128 * 128$ to $1024 * 1024$ and keeping all the other parameters constant. The algorithm was made to run on a Windows 10 Operating System computer with Intel Core i7-10510 CPU running at 1.8GHz (8 CPUs) and a 16 GB main memory. The plot of input size versus computation time is as shown in figure 7.26.

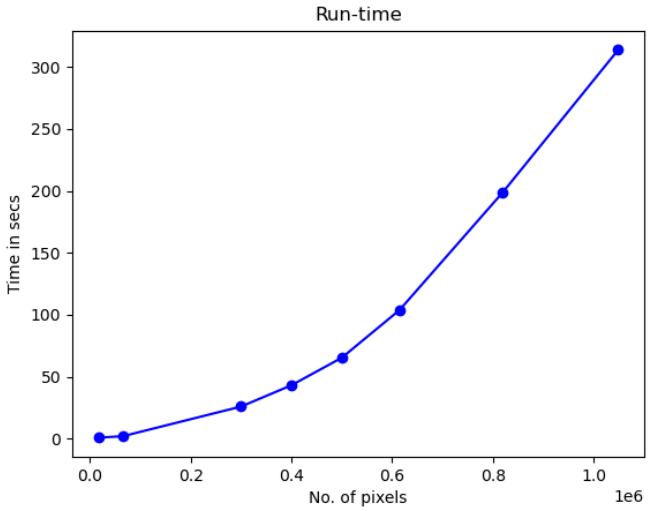

Figure 7.26: Plot of image size v/s time taken for Non-Local Self-Similarity based image denoising algorithm to complete execution

The plot in figure 7.26 does not include image sizes tending to infinity

(i.e., large input sizes), and hence the plot might not indicate the exact complexity class of the algorithm. However, by observing the plot, one can infer that the time required for computation increases sharply as the input size increases. The graph suggests that the algorithm belongs to a worse complexity class than linear time, i.e., $O(n)$. For images with size $1024 * 1024$, the algorithm took over 300 secs to run. Often the algorithm has to deal with image stacks containing multiple images. In such scenarios, the algorithm becomes unusable.

7.6.2 Slow sections of the algorithm

Python code was profiled [Kho21] to determine the execution time of various functions within the algorithm. With the help of profiling, even the total execution time of different functions that are present deep inside multiple functions can also be tracked [Kho21]. It was observed that the template matching algorithm took around 85-90% of the total run time of the denoising algorithm. Hence, template matching is the slowest section of the algorithm, especially since it is called multiple times with multiple *reference patches*. To make the algorithm more usable, the algorithm had to be optimized to enhance computation speed. Optimization methods and the results obtained are discussed in the next chapter.

It should be noted that the clustering step of the algorithm can also take much time when there are many samples. However, the sample size has been restricted in the algorithm (as discussed before).

Chapter 8

Optimization for run-time performance and denoising of multi-modal images

As discussed in the previous chapter, the template-matching algorithm makes the denoising process slow. Hence optimization methods were implemented to improve the run-time efficiency. The following is an explanation of the optimization methods and their results.

8.1 Optimization for run-time performance

8.1.1 Parallelization

Parallelization is the execution of computer programs to process the data at the same time. When a program is made to run in serial, the different tasks of the program run one after the other sequentially. But when a program is made to run in parallel, the different tasks of the program are made to run independently, often on different processors of the computer. The independent execution in parallel makes parallel computing faster. Parallelization is being used for many years especially in the field of supercomputing. It is a powerful technique to optimize the run-time efficiency of a program [Hop17].

Python uses something called Global Interpreter Lock (GIL), which is a type of processor lock. Because of GIL, multiple threads cannot access the same objects created in Python at a given point in time. Hence multi-threading in Python is not concurrent and cannot be used to optimize the speed in Python. However, the *multiprocessing*[1] library in Python side-steps the GIL by using subprocesses instead of threads. It makes use of the multiple processors in a computer to process data concurrently. The *multiprocessing* library runs both on Unix and Windows.

From the analysis in section 7.6, it is clear that the template-matching has a direct impact on the run-time efficiency of the denoising algorithm. From the flowchart of the algorithm (see figure 7.3), one can observe that the template matching algorithm runs multiple times as it is present in a loop. Even during every iteration of the loop, the template matching algorithm is called multiple times as there are multiple patches (i.e., *reference patches*) that are to be matched with multiple images. Hence in every iteration, for every image, every patch is matched by the application of the template matching algorithm, and the results obtained are used for classification. It should be noted that the template matching result with one template is not dependent on that of another template. The order of these template matching results is also not relevant for the next classification task. This means that the task is well suited for parallelization. Hence, the execution of the template matching algorithm was parallelized in every iteration.

To test the parallelized version of the algorithm, a stack of images was used instead of a single image. The image stack has multiple images of the same sample which are slightly misaligned. When these multiple images are used as an input at once, the patches from different images are matched which ensures that the information from different images are used for the denoising task. This improves the denoising quality as there is more information.

To measure the performance of the original and parallel version of the algorithm, both versions were made to run with similar parameters. In order to distinguish the differences in performance speed, a larger input size was

[1] https://docs.python.org/3/library/multiprocessing.html

used, since with a small input size the computation time is not a limiting factor. Furthermore, the additional overhead for the parallelization might have significant impact on the result for small input sizes. On running the parallelized version of the denoising algorithm with 10 images of size $1024 * 1024$, patch size of $46 * 46$, loop counter (see figure 7.3) of 5 and a maximum number of patches (see section 7.2) in a group as 400, the run time was around **764.38 secs (≈12 mins)**. Whereas the original version of the algorithm took **1626.31 secs (≈27 mins)**. The algorithms were made to run on a Windows 10 Operating System computer with Intel Core i7-10510 CPU (8 CPUs) and a 16 GB main memory.

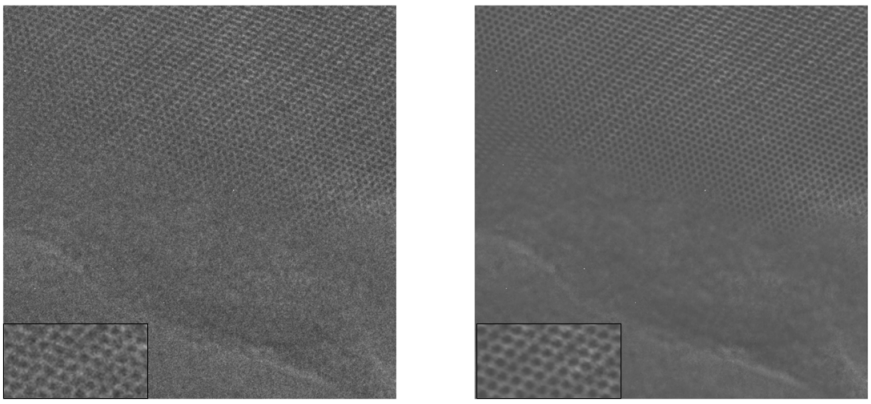

Figure 8.1: First image from the stack and its denoised result. Example of multi-slice denoising

On observing the computation times of the original and the parallelized versions of the algorithm, it can be observed that the parallelized version was almost twice as fast as the original version. This was on a typical personal computer. Although the parallelized version made use of all the 8 processors, the performance was improved only by about 2 fold. This is because that multiprocessing requires some extra time for initializing and allocating the processors, and the subtasks that are parallelized can have dependencies between them. Therefore, the entire period of the subtasks is not concurrent when they are running on different processes.

Figure 8.1 shows the first image in the stack and its corresponding de-

noised result. It can be observed that the result is as good as the result obtained from running the original version of the algorithm. Since the denoising algorithm uses information from other images as well, the denoising quality can be better with stacks of similar images. But in this case, the additional improvement is not evident. It should be noted that the functioning of the algorithm is same as the initial version, but the data is processed in a parallel way.

On observing the computation time of the original and the parallelized versions of the algorithm, it can be noted that the parallelized version provides a reasonable improvement in the performance without the requirement of powerful hardware. However, a run time of ≈12 mins for the parameters used is still a bit slow especially to apply the algorithm regularly. This motivated us to come up with an alternative solution for template matching, which is discussed in the next section.

8.1.2 Cosine similarity to improve run-time performance

It was observed that the template matching algorithm prevents large performance improvements of the denoising algorithm. Hence, it was required to replace the template matching algorithm with something that was faster. For this reason cosine similarity was implemented as an alternative to the previously used template matching algorithm.

Cosine Similarity is a measure that is used to quantify similarities between two vectors [Ala21] by finding the cosine of the angle between two vectors. If the vectors are in the same direction (i.e., similar), cosine Similarity is maximal. Cosine Similarity is defined as the normalized dot product of the two vectors. It is mathematically represented as,

$$similarity = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{8.1}$$

where A and B are two vectors, and θ is the angle between them. From the equation, it is obvious that the similarity value lies between 0 and 1. If we consider the two vectors to be images, convolution operation between an image and a patch results in a local dot product. Normalizing the result of

convolution between the image and the patch gives us the cosine similarity result of the patch (or the template) with patches at different locations of the image. Hence cosine similarity can be used to match patches with images, similar to template matching. An example of this application can be seen in [Ala21].

In the figure 8.2, an image and a patch taken from location (250,50) can be seen in the leftmost image. The result of applying the template matching algorithm can be seen in the middle image. The zoomed-in region shows the maximum at (250,50). Similarly, the result of applying cosine similarity and the maximum at (250,50) can be seen in the rightmost image. Observing these images, one can infer that the results of template matching and cosine similarity are very close to each other, which is the reason why the latter was chosen to replace the former.

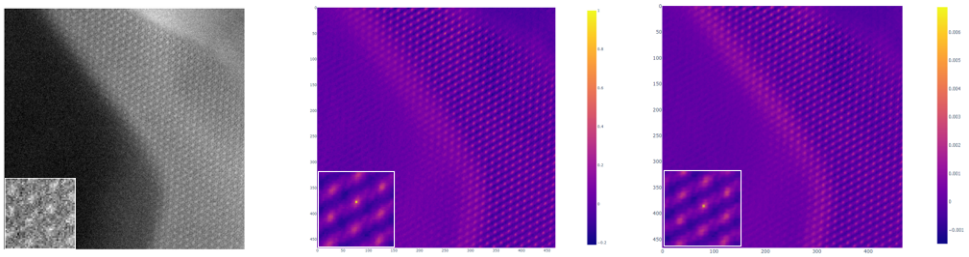

Figure 8.2: Image and a patch (leftmost), results of template matching algorithm (center) and cosine similarity (rightmost)

The time complexity of the convolution operation is $O(m^2n^2)$, where $m * m$ is the size of the template and $n * n$ is the size of the image, which is for large m worse than the run-time of the template matching algorithm. Complexity of the template matching algorithm is $O(n^2log(n^2))$ [Lew01], where $n * n$ is the data size. Since the convolution operation is widely used in CNNs, there are Python libraries that support GPU computation for performing the convolution operation. Running the computations on the GPU makes the algorithm much faster.

No significant change in the denoising quality between the template matching and the cosine similarity method was observed. For comparing

the run-time performance, the denoising task was carried out by running the original, parallelized and cosine similarity versions of the denoising algorithms with 10 images of size $1024 * 1024$, patch size of $46 * 46$, loop counter (see figure 7.3) of 15 and a maximum number of patches (see section 7.2) in a group as 400. On running the algorithms on a computer with Windows 10 OS, 128 GB of RAM, Intel Xenon processor (16 CPUs), and Nvidia RTX6000 GPU, the results obtained are tabulated in table 8.1.

Algorithm Version	**Computation time (secs)**
Original	4201.01 (≈70.01 mins)
Parallelized	1780.81 (≈29.7 mins)
Using Cosine Similarity	281.81 (≈4.7 mins)

Table 8.1: Computation time of different algorithm versions

On observing the results in table 8.1, it is evident that using Cosine Similarity provides a great advantage concerning speed. The only constraint here is that the computer needs to have a GPU for faster parallel computations.

8.2 Denoising of multi-modal images

In microscopy, there are different imaging modes. Often when an image of a specimen is captured, it is captured in different imaging modes. Since it is the same information in different modes, images in different modes can be used together for the denoising task.

When multi-modal images are used to be denoised, the different modes form the different channels of the input image. Thus when patches are extracted from the image, the patches have multiple channels too. But during the execution of the template matching or the cosine similarity algorithm, the patches and images in separate channels are treated differently i.e., as single $2D$ images and patches. The patches of n^{th} channel are matched with the images in the n^{th} channel. Thus n results are obtained for n channels. The results from the different channels are averaged to get one single result.

This is done for all multichannel images with each multichannel patch. Thus obtained results are used to classify the multi channel patches belonging to the multichannel images. The rest of the algorithm is the same but is carried out for multiple channels without the further exchange of data between channels.

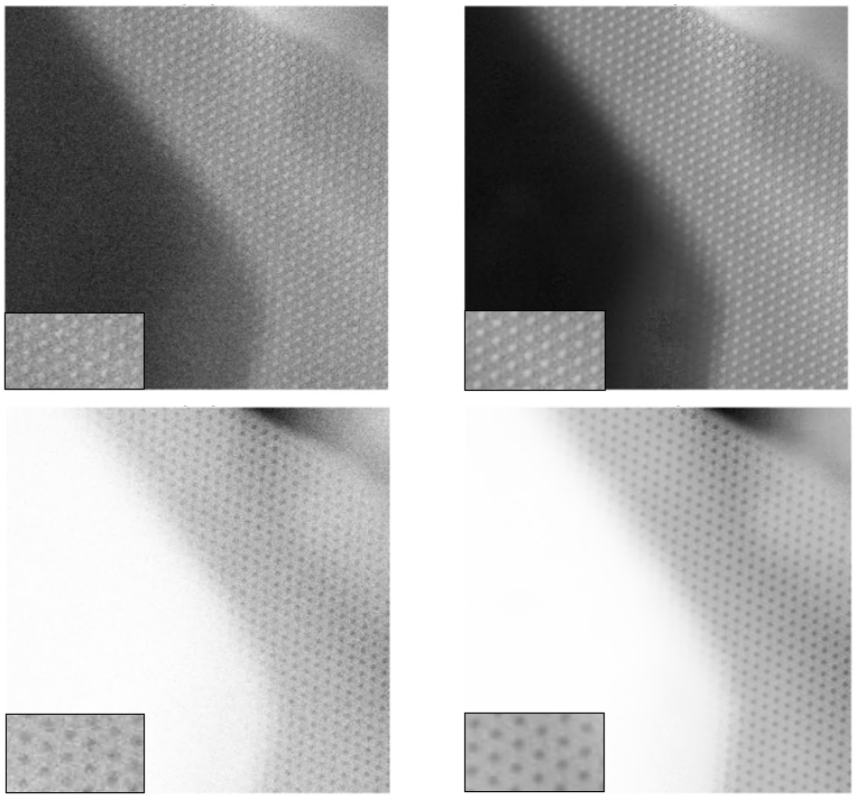

Figure 8.3: Noisy image in bright field mode and its result (top row). Noisy image in dark field mode and its result (bottom row)

Averaging of the template matching or the cosine similarity results is useful in extracting information from different channels. It is especially useful when the image in one mode is significantly more corrupted than the images in other modes. The results of running the algorithm with multimodal data is as shown in figure 8.3. In this case, the results are as at least good as those obtained with the original version of the algorithm.

Chapter 9

Future Work and Summary

9.1 Future Work

During the development and analysis of the new algorithm, several ideas for future improvements were found. In the following, two of these ideas are presented.

9.1.1 Application of transfer learning

Transfer learning is a deep learning method which uses a pre-trained model. In transfer learning, a model developed for one task is reused and fine tuned for another task. Such models will have the capability to differentiate wide range of images, as the data used to train these models are diverse. For instance, ImageNet [1] is a dataset which contains around 1.2 million images with 1000 categories. Networks like VGG16, ResNet50 and Xception are trained using the ImageNet dataset [Ros17].

In the Non-Local Self-Similarity based image denoising algorithm, template matching is applied with different reference patches. The template-matching result is then used to classify patches at different positions to form groups, that represented by *reference patches*. This process can be seen as a machine learning classification problem and transfer learning can be used to do the same. Assume that an image consists of n distinctive regions, so

[1] https://image-net.org/index.php

any patch in an image should belong to one of these n regions. Therefore, in this case n can be considered as the number of classes for classification.

The research of [CH21] is an example of transfer learning being applied to classify images without ground truth data. In the case of our denoising algorithm, patches can be extracted from images, which become the input for the pre-trained model. From the pre-trained network, the patch features can be obtained by removing the last fully connected layer. Now the patches are represented by features. If the feature size is too large, Principle Component Analysis can be applied to reduce the feature size and consider only few important features. Clustering methods can be applied on the features to group similar patches. This way similar patches can be found and averaged. Using pre-trained networks can be faster than running the template matching algorithm multiple times.

Classification of image patches with transfer learning was experimented with the VGG16 network. This was partially successful as the patches within a class were similar but they were not similar enough that they produced artifacts when they were averaged. Hence the application of transfer learning has to be further improved and be made to produce robust results. This could be a future work that can potentially improve the performance of denoising without compromising on the quality of denoising.

9.1.2 Ensemble denoising

From running the Non-local Self-Similarity based denoising algorithm multiple times, it was observed that there can be minute errors. The reason for this can be the way how initial *reference patches* are selected or problems during the clustering when there is a single cluster formed with dissimilar patches. An improvement for the current algorithm can be made where the algorithm is made to run n times with different initial positions and clustering parameter values. This produces n denoised results, which can be averaged to get a final denoised result. Another option would be to use the confidence map of n denoised results and build a final denoised image based on the confidence maps of these n results.

9.2 Summary

Denoising of the experimental images obtained from X-ray and electron microscopy was performed by using different denoising methods - generic CNN based auto-encoder, Block Matching and 3D filtering (BM3D) and Noise2Void (N2V). The denoised results obtained from the auto-encoder was of an average quality. But BM3D and N2V did a decent job in performing the denoising task. The level of denoising was not sufficient to clearly enhance the minute structures present in the experimental images.

A new denoising algorithm was developed which works based on the similar structures present in images and averaging them to suppress the noise. This method successfully denoises the images and also enhances the minute structures which was not possible by the other methods. The proposed algorithm was analyzed and it's behavior under different circumstances was explained. The results of all the denoising methods were compared using Fourier image analysis. The new denoising algorithm was also tested on natural images represented by 8 bit data format. The algorithm performs well on natural images too. Upon comparing the PSNR values of the denoised results using test images, it was clear that the new algorithm outperformed all the previously tested methods.

However, the original version of the new denoising algorithm was not well suited for denoising image stacks involving multiple images. This was due to the long run-time of the algorithm. Therefore, the algorithm was optimized to enhance the speed of computation by introducing cosine similarity instead of template matching. This helped in enhancing the run-time efficiency by a factor of 14.5 approximately, for the hardware used in our experiments. Additionally, the new algorithm was extended to perform the denoising of multi-modal images.

Appendix A

Algorithms

Algorithm 1 Denoising Algorithm

Input: Noisy image
Result: Denoised image
imgs ← loadImgs() ;
set [*templateSize, minGroupSize, maxGroupSize, clustParam*] ;
refPatches ← generate_initial_ref_patches(*imgs*, *templateSize*) ;
$c \leftarrow 15$;
for $c \geq 0$ **or** (check if the number of *refPatches* in the last two iterations has changed) **do**
    tmpMatchResult ← [ ] ;
    for *img* in *imgs* **do**
        for *refPatch* in *refPatches* **do**
            tmpMatchResult.append(template_matching(*img*, *refPatch*)) ;
        end for
    end for
    maxValue ← max(*tmpMatchResult*, axis=2) ;
    refPatchIndex ← '*refPatches*' index at *maxValue* ;
    [*sorted_maxValue, positionX, positionY*] ← sortWithIdx(*maxValue*);
    t = *templateSize* ;
    list[idx,posX,posY] = [ ];

```
    for each s,posX,posY in sorted_maxValue, positionX, positionY do
        if maxValue[posX,posY] not equal 0 then
            maxValue[posX: posX+(t/4); posY: posY+(t/4)] ← 0 ;
            idx ← refPatchIndex[posX,posY] ;
            list.append([idx,posX,posY]) ;
        end if
    end for
    [count,refPatchID] ← count_patches_with_same_id(list);
    for cnt,ID in count,refPatchID do
        if cnt ≤ minGroupSize then
            list ← delete(list, ID);
        else if cnt ≥ maxGroupSize then
            list ← splitGroup(list, ID, maxGroupSize);
        end if
    end for
    refPatches ← average_patches_with_same_id(list) ;
    c ← c-1 ;
end for
subGroup[centroid, posX, posY] = [ ];
for refPatchID in refPatches do
    patches_with_sameRef ← get_patches_with_same_id(list,refPatchID);
    numClusters ← count(patches_with_sameRef)/clustParam ;
    subGroup.append(cluster(patches_with_sameRef, numClusters)) ;
end for
gaussianWeight ← createGaussianWeight();
denoisedImg ← backPlot(subGroup, gaussianWeight);
```

Algorithm 2 Algorithm for calculating variance

1: $weight \leftarrow weightA + weightB$;
2: $delta \leftarrow avgB$ - $avgA$;
3: $M2 \leftarrow (M2\_A + M2\_B + delta^2 * weightA * weightB)$ / n ;
4: $Var\_AB = M2 / (n - 1)$;

Appendix B

Other results

Following are the images and their corresponding results by applying the Non-Local Self-Similarity based image denoising algorithm on other electron microscopy images.

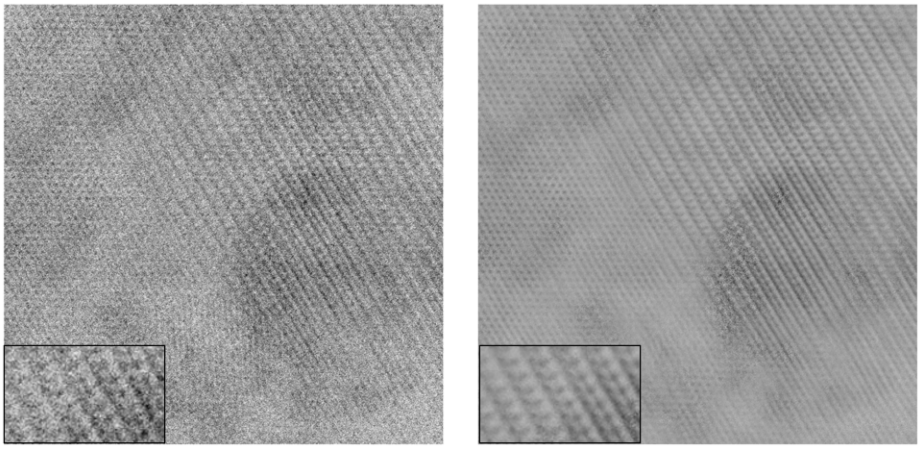

Figure B.1: Noisy image-1 and it's denoised result

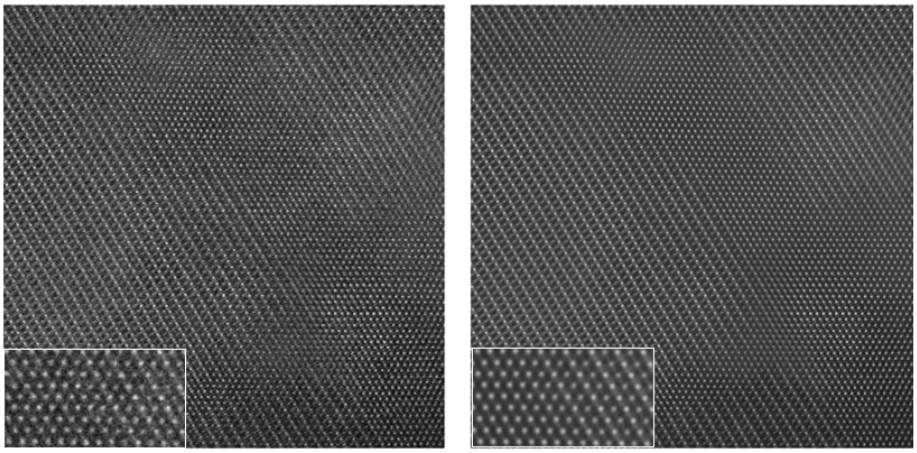

Figure B.2: Noisy image-2 and it's denoised result

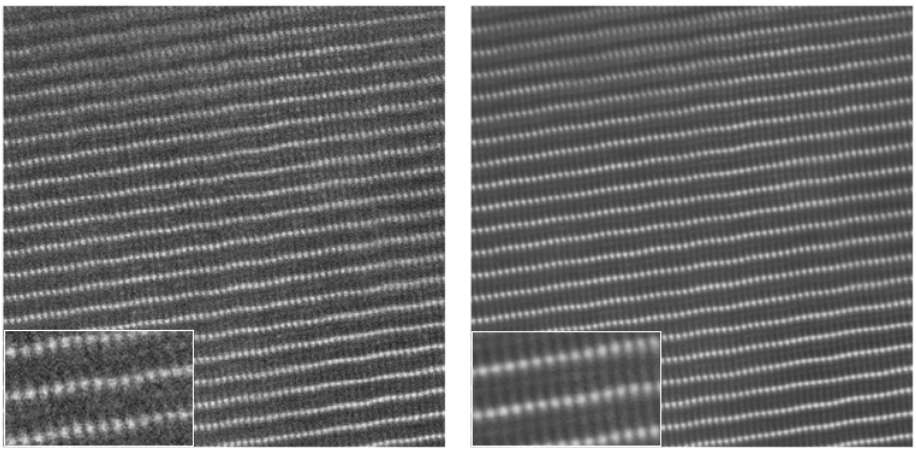

Figure B.3: Noisy image-3 and it's denoised result

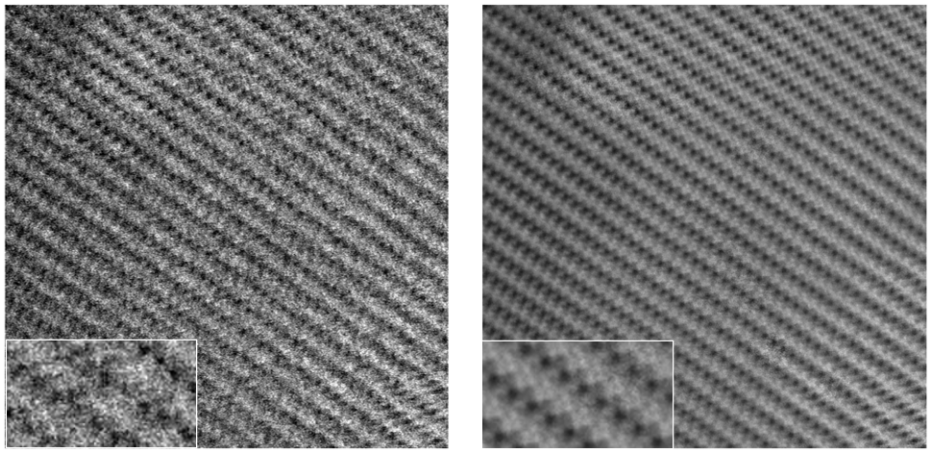

Figure B.4: Noisy image-4 and it's denoised result

Bibliography

[AC17] Byeongyong Ahn and Nam Ik Cho. Block-matching convolutional neural network for image denoising. *CoRR*, abs/1704.00524, 2017.

[Ada20] Nikolas Adaloglou. Intuitive explanation of skip connections in deep learning. *https://theaisummer.com/*, 2020.

[Ala21] Richmond Alake. Understanding Cosine Similarity And Its Application, September 2021.

[BCM05] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, 2005.

[BCM11] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-Local Means Denoising. *Image Processing On Line*, 1:208–212, 2011.

[Ben09] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

[Bhu17] Shashi Bhushan. Single Particle Reconstruction, December 2017.

[Bou20] Louis Bouchard. What is Self-Supervised Learning? | Will machines ever be able to learn like humans?, May 2020.

[BR19] Joshua Batson and Loïc Royer. Noise2self: Blind denoising by self-supervision. *CoRR*, abs/1901.11365, 2019.

[BT92] Luigi Bedini and Anna Tonazzini. Image restoration preserving discontinuities: the bayesian approach and neural networks. *Image and Vision Computing*, 10(2):108–118, 1992.

[Cat12] Philippe Cattin. Image Restoration: Introduction to Signal and Image Processing, April 2012.

[CCD08] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. *Supervised Learning*, pages 21–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[CGL82] Tony F Chan, Gene H Golub, and Randall J LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, pages 30–41. Springer, 1982.

[CH21] Ryan Cohn and Elizabeth Holm. Unsupervised machine learning via transfer learning and k-means clustering to classify materials image data. *Integrating Materials and Manufacturing Innovation*, 10(2):231–244, Apr 2021.

[Cho16] Francois Chollet. Building Autoencoders in Keras, May 2016.

[CM05] Bartomeu Coll and Jean-Michel Morel. Image denoising by non-local averaging. volume 2, pages 25– 28, 02 2005.

[CS89] Yi-Wu Chiang and Barry J. Sullivan. Multi-frame image restoration using a neural network. pages 744–747, 1989. cited By 6.

[DFKE06] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising with block-matching and 3D filtering. In Nasser M. Nasrabadi, Syed A. Rizvi, Edward R. Dougherty, Jaakko T. Astola, and Karen O. Egiazarian, editors,

Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning, volume 6064, pages 354 – 365. International Society for Optics and Photonics, SPIE, 2006.

[DFKE07] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[EP16] Alfredo E. Palacios Enríquez and Volodymyr Ponomaryov. Image denoising using block matching and discrete cosine transform with edge restoring. In *2016 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 140–147, 2016.

[Fan00] Engui Fan. Extended tanh-function method and its applications to nonlinear equations. *Physics Letters A*, 277(4):212–218, 2000.

[Fra96] Joachim Frank. Three-dimensional electron microscopy of macromolecular assemblies: Visualization of biological molecules in their native state. 1996.

[FZFZ19] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. Brief review of image denoising techniques. *Vis. Comput. Ind. Biomed. Art*, 2:1–12, 2019.

[GAAB18] Maryam Gholizadeh-Ansari, Javad Alirezaie, and Paul Babyn. Low-dose ct denoising with dilated residual network. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5117–5120, 2018.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Gro17] Prince Grover. Various Implementations of Collaborative Filtering, December 2017.

[GRP15] S. Gopinathan, Kokila Radhakrishnan, and Thangavel P. Wavelet and fft based image denoising using non-linear filters. *International Journal of Electrical and Computer Engineering (IJECE)*, 5:1018–1026, 10 2015.

[GW+02] Rafael C Gonzalez, Richard E Woods, et al. Digital image processing, 2002.

[GW06] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., USA, 2006.

[Hop17] Computer Hope. Parallelization, April 2017.

[Hos07] WJ Hossack. Noise in images - the university of edinburgh, Aug 2007.

[Hou07] JH Hou. *Research on image denoising approach based on wavelet and its statistical characteristics*. PhD thesis, Dissertation, Huazhong University of Science and Technology, 2007.

[HPB20] Allard Adriaan Hendriksen, Daniël Maria Pelt, and K. Joost Batenburg. Noise2inverse: Self-supervised deep convolutional denoising for tomography. *IEEE Transactions on Computational Imaging*, 6:1320–1335, 2020.

[HTG08] Q. Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics Letters*, 44:800 – 801, 02 2008.

[HUA71] TS HUANG. Stability of two-dimensional recursive filters(mathematical model for stability problem in two-dimensional recursive filtering). 1971.

[HY18] Y. Han and J.C. Ye. Framing u-net via deep convolutional framelets: Application to sparse-view ct. *IEEE Transactions on Medical Imaging*, 37(6):1418–1429, 2018. cited By 194.

[JBC+21] Erik K. Johnson, Stephen Becker, Carol Cogswell, Jian Xing, and Simeng Chen. Limitations of Fourier ring correlation as an image resolution metric. In Thomas G. Brown, Tony Wilson, and Laura Waller, editors, *Three-Dimensional and Multidimensional Microscopy: Image Acquisition and Processing XXVIII*, volume 11649. International Society for Optics and Photonics, SPIE, 2021.

[JHWL08] Licheng Jiao, Biao Hou, Shuang Wang, and Fang Liu. Image multiscale geometric analysis: Theory and applications beyond wavelets. *Xi'an Univ. Electron. Sci. Technol*, 1:124–132, 2008.

[JJR+19] Worku Jifara, Feng Jiang, Seungmin Rho, Maowei Cheng, and Shaohui Liu. Medical image denoising using convolutional neural network: a residual learning approach. *The Journal of Supercomputing*, 75(2):704–718, February 2019.

[KBJ19] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2019.

[Kho21] Ehsan Khodabandeh. How to Profile Your Code in Python, June 2021.

[Kim13] Oliver Kim. What are the differences between brightfield, darkfield and phase contrast? | Microbehunter Microscopy, February 2013.

[Kna13] Claude Knaus. Dual-image denoising. Technical report, Universität Bern, Bern, November 2013.

[KTC+19] Sami Koho, Giorgio Tortarolo, Marco Castello, Takahiro Deguchi, Alberto Diaspro, and Giuseppe Vicidomini. Fourier ring correlation simplifies image restoration in fluorescence microscopy. *Nature communications*, 10(1):1–9, 2019.

[KVP+20] Alexander Krull, Tomáš Vičar, Mangal Prakash, Manan Lalit, and Florian Jug. Probabilistic noise2void: Unsupervised content-aware denoising. *Frontiers in Computer Science*, 2:5, 2020.

[Lar11] Schmidt-Thieme Lars. Image Analysis - Fourier Transform, 2011.

[LC10] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[Lef17] Stamatios Lefkimmiatis. Universal denoising networks : A novel cnn-based network architecture for image denoising. *CoRR*, abs/1711.07807, 2017.

[Lew01] J.P. Lewis. Fast normalized cross-correlation. *Ind. Light Magic*, 10, 10 2001.

[LKLA19] Samuli Laine, Tero Karras, Jaakko Lehtinen, and Timo Aila. High-quality self-supervised deep image denoising. *Advances in Neural Information Processing Systems*, 32:6970–6980, 2019.

[LMH+18] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. *CoRR*, abs/1803.04189, 2018.

[LYC+17] Donghoon Lee, Sangdoo Yun, Sungjoon Choi, Hwiyeon Yoo, Ming-Hsuan Yang, and Songhwai Oh. Unsupervised holistic image generation from key local patches. *CoRR*, abs/1703.10730, 2017.

[Mac06] L.W. MacDonald. *Digital Heritage: Applying Digital Imaging to Cultural Heritage*. Elsevier, 2006.

[MAF20] Ymir Mäkinen, Lucio Azzari, and Alessandro Foi. Collaborative filtering of correlated noise: Exact transform-domain vari-

ance for improved shrinkage and patch matching. *IEEE Trans. Image Process.*, 29:8339–8354, 2020.

[mDDS+15] İmren Dinç, Semih Dinç, Madhav Sigdel, Madhu S. Sigdel, Ramazan S. Aygün, and Marc L. Pusey. Chapter 12 - dt-binarize: A decision tree based binarization for protein crystal images. In Leonidas Deligiannidis and Hamid R. Arabnia, editors, *Emerging Trends in Image Processing, Computer Vision and Pattern Recognition*, pages 183–199. Morgan Kaufmann, Boston, 2015.

[MDKF08] André C. Marreiros, Jean Daunizeau, Stefan J. Kiebel, and Karl J. Friston. Population dynamics: Variance and the sigmoid activation function. *NeuroImage*, 42(1):147–157, 2008.

[MDP14] Naresh Marturi, Sounkalo Dembélé, and Nadine Piat. Scanning electron microscope image signal-to-noise ratio monitoring for micro-nanomanipulation. *Scanning*, 36, 07 2014.

[MGMH04] Mukesh Motwani, Mukesh Gadiya, Rakhi Motwani, and Frederick Harris. Survey of image denoising techniques. 01 2004.

[Mig07] Max Mignotte. Image denoising by averaging of piecewise constant simulations of image partitions. *IEEE Transactions on Image Processing*, 16(2):523–533, 2007.

[MSY16] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. 06 2016.

[Ng17] Andrew Ng. Unsupervised Feature Learning and Deep Learning Tutorial, December 2017.

[noa20] Optimizing Image Signal to Noise Ratio Using Frame Averaging, October 2020.

[OK11] A Olgac and Bekir Karlik. Performance analysis of various activation functions in generalized mlp architectures of neural

networks. *International Journal of Artificial Intelligence And Expert Systems*, 1:111–122, 02 2011.

[PKJ21] Mangal Prakash, Alexander Krull, and Florian Jug. Fully unsupervised diversity denoising with convolutional variational autoencoders. In *International Conference on Learning Representations*, 2021.

[PN15] Jean-François Pambrun and Rita Noumeir. Limitations of the ssim quality metric in the context of diagnostic imaging. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 2960–2963, 2015.

[PV13] Ioannis Pitas and Anastasios N Venetsanopoulos. *Nonlinear digital filters: principles and applications*, volume 84. Springer Science & Business Media, 2013.

[Rei84] Ludwig Reimer. *Transmission electron microscopy : physics of image formation and microanalysis / Ludwig Reimer*. Springer-Verlag Berlin ; New York, 1984.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[Ros77] A. Rose. *Vision: human and electronic*. Optical physics and engineering. Plenum Press, 1977.

[Ros17] Adrian Rosebrock. ImageNet: VGGNet, ResNet, Inception, and Xception with Keras, March 2017.

[SAU19] Umme Sara, Morium Akter, and Mohammad Uddin. Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 07:8–18, 01 2019.

[Sha19] Pulkit Sharma. A Beginner's Guide to Hierarchical Clustering and how to Perform it in Python, May 2019.

[Sin21] Himanshi Singh. Deep Learning 101: Beginners Guide to Neural Network, March 2021.

[Sip13] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, Boston, MA, third edition, 2013.

[Ste97] W. Smith Steven. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub, first edition, January 1997.

[STP04] K.S. Sim, J.T.L. Thong, and J.C.H. Phang. Effect of shot noise and secondary emission noise in scanning electron microscope images, 2004.

[Sus] Igor Susmelj. The Advantage of Self-Supervised Learning.

[TFZ+20] Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin. Deep learning on image denoising: An overview. *Neural Networks*, 131:251–275, 2020.

[TM98] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, 1998.

[Tsa18] Sik-Ho Tsang. Review: DilatedNet — Dilated Convolution (Semantic Segmentation), November 2018.

[Urb17] Kevin Urban. Notes on Autoencoders (Decoded into English), 2017.

[WBSS04] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[Wik21] Wikipedia contributors. Single particle analysis — Wikipedia, the free encyclopedia, 2021. [Online; accessed 27-November-2021].

[WSB+18] Martin Weigert, Uwe Schmidt, Tobias Boothe, A. Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, S. Culley, Maurício Rocha-Martins, Fabián Segovia-Miranda, C. Norden, R. Henriques, M. Zerial, M. Solimena, J. Rink, P. Tomançak, Loic A. Royer, F. Jug, and E. Myers. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature Methods*, 15:1090–1097, 2018.

[XWJ20] Yaochen Xie, Zhengyang Wang, and Shuiwang Ji. Noise2same: Optimizing A self-supervised bound for image denoising. *CoRR*, abs/2010.11971, 2020.

[YLZ+18] Yuan Yuan, Siyuan Liu, Jiawei Zhang, Yongbing Zhang, Chao Dong, and Liang Lin. Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks. *CoRR*, abs/1809.00437, 2018.

[YYG+95] Ruikang Yang, Lin Yin, Moncef Gabbouj, Jaakko Astola, and Yrjö Neuvo. Optimal weighted median filtering under structural constraints. *IEEE transactions on signal processing*, 43(3):591–604, 1995.

[ZBW05] Lei Zhang, Paul Bao, and Xiaolin Wu. Multiscale lmmse-based image denoising with optimal wavelet selection. *IEEE Transactions on circuits and systems for video technology*, 15(4):469–481, 2005.

[ZOKB19] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S^4l: Self-supervised semi-supervised learning. *CoRR*, abs/1905.03670, 2019.

[ZZC+17] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

[ZZZ18] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.