

**INVERSE PROBLEM IN IMAGE PROCESSING:
IMAGE RESTORATION**

WONG YEN KHAI

UNIVERSITI TUNKU ABDUL RAHMAN

**INVERSE PROBLEM IN IMAGE PROCESSING:
IMAGE PROCESSING**

WONG YEN KHAI


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Electrical and Electronic
Engineering with Honours**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

May 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 
Name : Wong Yen Khai
ID No. : 1801880
Date : 12/5/2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**INVERSE PROBLEM IN IMAGE PROCESSING: IMAGE RESTORATION**” was prepared by **WONG YEN KHAI** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Electrical and Electronic Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature :  _____

Supervisor : Chua Sing Yee _____

Date : 15 May 2023 _____

Signature : _____

Co-Supervisor : _____

Date : _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2023, Wong Yen Khai. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to take this opportunity to express my sincere appreciation to everyone who contributed to the successful completion of this project. I am especially grateful to my research supervisor, Dr. Chua Sing Yee, for her invaluable guidance, advice, and patience throughout the research process. Her expert feedback and constructive criticism were instrumental in shaping the direction and scope of this project, and her unwavering support and encouragement helped me stay motivated during the challenging times.

In addition, I would also like to express my gratitude to my parents and friends who have been a constant source of support and encouragement throughout this journey. Their belief in my abilities and their unwavering support have been a great source of strength and inspiration for me, and I am deeply grateful for their contributions to this project.

ABSTRACT

Conventional image restoration methods often require careful feature selection and fine-tuning, which can be a complicated process and not always possible. On the other hand, Deep-learning (DL) models rely heavily on the datasets availability and neural network architecture, which can lead to reduced performance if the network is poorly designed. Recently, Deep Image Prior (DIP), a learning-free approach to image restoration has emerged as an alternative. However, DIP requires a pre-defined early stopping, which can limit its practical applications. Hence, this project aims to improve image restoration through DIP and address the limitation mentioned. This research proposes the use of Metric-based Early Stopping (MB-ES) with the DIP model for image denoising and super-resolution tasks. The proposed MB-ES algorithm utilizes intermediate restored images to identify the optimal stopping point using PSNR and SSIM metrics, thus eliminating the need for pre-defined early stopping. The results show that MB-ES requires fewer iterations to obtain a better-quality image and has lower design complexity as compared to the existing Early Stopping using Exponential Moving Variance (ES-EMV). The proposed MB-ES algorithm with DIP is then evaluated on both image denoising and super-resolution tasks, and compared with classical and deep learning-based methods. The results show that the proposed MB-ES algorithm achieves remarkable performance in detecting the stopping point that closely resembles the ground truth. In general, the proposed MB-ES on DIP outperforms classical methods and shows comparable performance with recent deep-learning-based models. It is worth noting that DIP does not require heavy training on massive datasets to achieve the performance that DL models possess. The findings of this research are hoped to benefit practical applications especially when dataset is not available and computational resource is limited for DL.

TABLE OF CONTENTS

DECLARATION		i
APPROVAL FOR SUBMISSION		ii
ACKNOWLEDGEMENTS		iv
ABSTRACT		v
TABLE OF CONTENTS		vi
LIST OF TABLES		ix
LIST OF FIGURES		xi
LIST OF SYMBOLS / ABBREVIATIONS		xvi
LIST OF APPENDICES		xix
CHAPTER		
1	INTRODUCTION	1
1.1	General Introduction	1
1.2	Importance of the Study	1
1.3	Problem Statement	2
1.4	Aim and Objectives	3
1.5	Scope and Limitation of the Study	3
1.6	Contribution of the Study	4
1.7	Outline of the Report	4
2	LITERATURE REVIEW	5
2.1	Introduction	5
2.2	Basic Types of Images	6
2.2.1	Binary Images	7
2.2.2	Grayscale Images	7
2.2.3	Colour Images	8
2.3	Image Degradation Models	8
2.3.1	Blur Models	8
2.3.2	Noise Models	10
2.4	Classical Image Restoration Methods	13

2.4.1	Linear Filter	13
2.4.2	Non-linear Filter	16
2.4.3	Non-blind Deconvolution for Image Restoration	20
2.4.4	Blind Deconvolution Image Restoration	24
2.4.5	Investigation of Classical Image Restoration Methods	26
2.4.6	Summary	30
2.5	Deep Learning (DL) Approaches to Image Restoration	30
2.5.1	Image Deblurring	30
2.5.2	Image Denoising	35
2.5.3	Super-Resolution	38
2.5.4	Investigation of Deep Learning Approaches to Image Restoration	41
2.5.5	Summary	44
2.6	Deep Image Prior (DIP) Approaches to Image Restoration	44
2.6.1	Overview	44
2.6.2	Investigation of Deep Image Prior to Image Restoration	46
2.6.3	Summary	49
2.7	Overall Summary	49
3	METHODOLOGY AND WORK PLAN	50
3.1	Introduction	50
3.2	Work Plan	50
3.3	Implementation	53
3.4	Performance Metrics	54
3.5	Test Image Preparation	57
3.5.1	Denoising	57
3.5.2	Super-resolution	59
3.6	DIP Model	60
3.7	Early Stopping Detection Method	61
3.8	Proposed Method	61

	3.8.1 Denoising	63
	3.8.2 Super-Resolution	71
4	RESULTS AND DISCUSSION	75
	4.1 Introduction	75
	4.2 Investigation of Fine-tuning Parameter Values for the Proposed MB-ES	75
	4.2.1 Denoising-MB-ES Algorithm	76
	4.2.2 Super-Resolution-MB-ES Algorithm	76
	4.3 Comparison of the Proposed MB-ES (PSNR) and ES-EMV (Variance)	79
	4.4 Comparison of the Proposed Denoising-MB-ES and Other Denoising Models	82
	4.5 Comparison of the Proposed Super-Resolution- MB-ES and Other Super-Resolution Models	86
5	CONCLUSIONS AND RECOMMENDATIONS	92
	5.1 Conclusions	92
	5.2 Recommendations for future work	93
	REFERENCES	94
	APPENDICES	98

LIST OF TABLES

Table 2.1:	Evaluation Metrics on Restoration Outputs (Source: Navaneethakrishnan, 2014). Best score is underlined.	25
Table 2.2:	Performance Comparison on Cameraman Image Using Classical Methods.	27
Table 2.3:	Performance Comparison on Cameraman Image Using Deconvolution Methods.	29
Table 2.4:	Comparison of Image Deblurring Methods.	34
Table 2.5:	Result of PSNR Performance between FFDNet with Different Models on BSD68 Datasets (Source: Zhang et al., 2018; Isogawa et al., 2018).	36
Table 2.6:	Comparison of Image Denoising Methods.	37
Table 2.7:	Comparison of Super-resolution Methods.	40
Table 2.8:	Performance Comparison on CSet9 Datasets for Deblurring Results Using DL Approaches.	41
Table 2.9:	Performance Comparison on CSet9 Datasets for Denoising Results Using DL Approaches.	43
Table 2.10:	Performance Comparison on CSet9 Datasets for Super-Resolution Results Using DL Approaches.	44
Table 2.11:	Performance Comparison on CSet9 Datasets for Denoising Results Using DIP Model.	46
Table 2.12:	Performance Comparison on CSet9 Datasets for Super-Resolution Results using DIP Model.	47
Table 2.13:	Performance Comparison on CSet9 Datasets for Impainting Results using DIP Model.	49
Table 3.1:	Specifications of Local Platform and Google Colab in the Project.	54
Table 3.2:	Evaluation Metrics of Image Quality.	55
Table 3.3:	Parameters for Different Types of Noise.	58
Table 3.4:	Metrics Equation for Plotting Curves.	64

Table 3.5:	The Proposed Denoising-MB-ES Algorithm Based on PSNR and SSIM Methods on Lena Image.	69
Table 3.6:	Comparison of PSNR and SSIM Deviations for the Proposed Denoising-MB-ES Using PSNR and SSIM Method.	71
Table 4.1:	Comparison of PSNR and SSIM Deviations Between the Proposed MB-ES and ES-EMV.	81
Table 4.2:	Comparison of the Proposed MB-ES and ES-EMV.	82

LIST OF FIGURES

Figure 2.1:	Overview of Images Types and Image Degradation Models.	5
Figure 2.2:	Overview of Image Restoration Methods.	6
Figure 2.3:	Sample of Image Models.	7
Figure 2.4:	Sample of Blurring Models.	9
Figure 2.5:	Sample of Different Types of Noise Models.	11
Figure 2.6:	Graphical Representation of PDF for Gaussian Noise (Source: Kanrar and Maji, 2022).	12
Figure 2.7:	Graphical Representation of PDF for Speckle Noise (Source: Kanrar and Maji, 2022).	13
Figure 2.8:	Denoising Outputs Using Linear Filters.	14
Figure 2.9:	Convolution Process of Average Filter.	14
Figure 2.10:	Convolution Process of Gaussian Filter.	15
Figure 2.11:	Denoising Outputs Using Minimum and Maximum Filters.	16
Figure 2.12:	Calculation of Median Value with 3×3 Kernel.	17
Figure 2.13:	Denoising Output Using Median Filter to Remove Salt-and-Pepper Noise.	17
Figure 2.14:	Comparison between the Performance of Conservative Filter and Median Filter due to Different Salt-and-Pepper Noise Levels.	19
Figure 2.15:	Removal of Gaussian Noise with Bilateral Filter.	20
Figure 2.16:	Blurring of an Image with a Given Random PSF.	21
Figure 2.17:	Example of Output Using Lucy-Richardson Deconvolution for (a) Blurred Image, and (b) Noisy Blurred Image.	22
Figure 2.18:	Example of Output from Wiener Filter Deconvolution.	23
Figure 2.19:	Example of Output from Regularized Filtering Deconvolution.	24

Figure 2.20:	Salt-and-Pepper Denoising Results on Cameraman Image Using Classical Methods.	26
Figure 2.21:	Gaussian Denoising Results on Cameraman Image Using Classical Methods.	27
Figure 2.22:	Restoration Results on Blurred Cameraman Image Using Deconvolution Methods.	28
Figure 2.23:	Restoration Results on Blurred and Noisy Cameraman Image Using Deconvolution Methods.	29
Figure 2.24:	Multi-Scale Network.	31
Figure 2.25:	Sample of Deblurring Results of DL Approaches on CSet9 Datasets.	41
Figure 2.26:	Sample of Denoising Results of DL Approaches on CSet9 Datasets.	42
Figure 2.27:	Sample of Super-Resolution Results of DL Approaches on CSet9 Datasets.	43
Figure 2.28:	Overview of Deep Image Prior Model Network (Source: Ulyanov et al., 2018).	45
Figure 2.29:	Sample of Denoising Results of DIP Model on CSet9 Datasets.	46
Figure 2.30:	Sample of Super-Resolution Results of DIP Model on CSet9 Datasets.	47
Figure 2.31:	Sample of Impainting Results of DIP Model on CSet9 Datasets.	48
Figure 3.1:	General Process Flow.	51
Figure 3.2:	FYP1 Gantt Chart.	52
Figure 3.3:	FYP2 Gantt Chart.	53
Figure 3.4:	Examples of Clean and Noisy Images with Different Types of Noise and Intensity Level: Gaussian, Speckle, and Shot Noises. Note: The performance metrics was measured in the form of (PSNR, SSIM, RMSE).	59
Figure 3.5:	Examples of Downsampled Images with Different Scaling Factors for Super-Resolution. Note: SF refer to Scaling Factor.	59

Figure 3.6:	Relationship Between MSE, PSNR, and Variance During Image Denoising.	63
Figure 3.7:	Relationship Between MSE_GT, PSNR_GT, and PSNR_INT during Image Denoising. Note: MSE_GT refer to mean squared error metric between restored images and ground truth.	64
Figure 3.8:	Flowchart of the Denoising-MB-ES Algorithm for Early Stopping Detection. Note: The “similarity” could be PSNR or SSIM metrics. The size of the window used to compute the slope of the records is determined by multiplying "wait_count" value with the "window_size_multiplier". For instance, if the window_size_multiplier" is set to 100, the window size will be increased for 100 for every "wait_count" steps (e.g., 100, 200, 300, etc.).	66
Figure 3.9:	Analysis of PSNR Deviation in the Proposed Denoising-MB-ES Algorithm with PSNR and SSIM Methods on CSet9 Images. Refer to Table A-1 for more details.	70
Figure 3.10:	Analysis of SSIM Deviation in the Proposed Denoising-MB-ES Algorithm with PSNR and SSIM Methods on CSet9 Images. Refer to Table A-1 for more details.	70
Figure 3.11:	Relationship Between PSNR_GT and PSNR_INT During Super-Resolution.	72
Figure 3.12:	Relationship Between SSIM_GT and SSIM_INT During Super-Resolution.	72
Figure 3.13:	Flowchart of the Proposed Super-Resolution-MB-ES Algorithm for Early Stopping Detection. Note: The “similarity” is SSIM metric for super-resolution.	74
Figure 4.1:	Relationship Between Window Size Multiplier and PSNR/SSIM Deviations and Average Number of Iterations.	76
Figure 4.2:	Relationship Between Slope Threshold Parameter and PSNR/SSIM Deviations and Average Number of Iterations (with Patience = 1000, Window = 1000).	77
Figure 4.3:	Relationship Between Patience Parameter and PSNR/SSIM Deviations and Average Number of Iterations (with Slope Threshold = 2×10^{-6} , Window = 1000).	78

- Figure 4.4: Relationship Between Window Parameter and PSNR/SSIM Deviations and Average Number of Iterations (with Slope Threshold = 2×10^{-6} , Patience = 700). 78
- Figure 4.5: Relationship Between Slope Threshold Parameter and PSNR/SSIM Deviations and Average Number of Iterations (with Patience = 700, Window = 1000). 79
- Figure 4.6: Comparison of PSNR Deviation Between the Proposed MB-ES with ES-EMV for the Denoising on CSet9 Images. Refer to Table A-2 for more details. 80
- Figure 4.7: Comparison of SSIM Deviation Between the Proposed MB-ES with ES-EMV for the Denoising on CSet9 Images. Refer to Table A-2 for more details. 81
- Figure 4.8: Comparison of Low-level Speckle Denoising Results on F16 image for DD*, SGLD* and DIP*: Analysis of PSNR_INT against (a) PSNR_GT and (b) SSIM_GT. Note: The superscript "*" indicates the MB-ES was applied to the model. Further details can be found in Table A-3. 83
- Figure 4.9: Comparison of High-level Speckle Denoising Results on F16 image for DD*, SGLD* and DIP*: Analysis of PSNR_INT Against (a) PSNR_GT and (b) SSIM_GT. Note: The superscript "*" indicates the MB-ES was applied to the model. Further details can be found in Table A-3. 84
- Figure 4.10: Denoising Results of DD*, SGLD* and DIP* on Low- and High-Level Speckle Noises Images: F16, Peppers and kodim03. Note: The superscript "*" indicates the MB-ES was applied to the model. The performance metrics was measured in the form of (PSNR, SSIM, RMSE). 86
- Figure 4.11: Comparison of PSNR for Super-Resolution Models on CSet9 Dataset with Scaling Factors 2, 3, and 4. Note: The superscript "*" indicates the MB-ES was applied to the model. Please refer to Table A-4 for further details. 87
- Figure 4.12: Comparison of SSIM for Super-Resolution Models on CSet9 Dataset with Scaling Factors 2, 3, and 4. Note: The superscript "*" indicates the MB-ES was applied to the model. Please refer to Table A-4 for further details. 87
- Figure 4.13: Super-Resolution Results of Bicubic, SRCNN, ESPCN and DIP* on Different Scaling Factors of House Image. Note: The superscript "*" indicates the MB-ES was

applied to the model. The performance metrics was measured in the form of (PSNR, SSIM, RMSE). 89

Figure 4.14: Super-Resolution Results of Bicubic, SRCNN, ESPCN and DIP* on Different Scaling Factors of Lena Image. Note: The superscript “*” indicates the MB-ES was applied to the model. The performance metrics was measured in the form of (PSNR, SSIM, RMSE). 90

Figure 4.15: Super-Resolution Results of Bicubic, SRCNN, ESPCN and DIP* on Different Scaling Factors of Baboon Image. Note: The superscript “*” indicates the MB-ES was applied to the model. The performance metrics was measured in the form of (PSNR, SSIM, RMSE). 91

LIST OF SYMBOLS / ABBREVIATIONS

B_S	Bits per Sample
G_θ	Generator Function
$H_{i,j}$	Height of the Feature Map
I^B	Blurred Image
I^S	Smooth Image
L_{adv}	Local and Global Discriminator Losses
L_p	MSE Loss
L_x	Perceptual Loss
M_B	Performance Metric Score based on Best Recovered Image with respect to Ground Truth
M_D	Performance Metric Score based on Detected Recovered Image with respect to Ground Truth
$R_{dynamic}$	Pixel-Value Dynamic Range
$W_{i,j}$	Width of the Feature Map
d_j	Observed Value at Position- i
f_W	Function of Wiener Filter
m_f	Mean of Function
p_{ij}	PSF from the True Location of Position- j Relative to the Observation Position- i
u_j	Latent Image at Position- j
x_p	Image Pixels
σ^2	Variance of Specific Window or Blur
σ_N	Standard Deviation for Noise
A	Noisy Image
B	Original Image
B'	Estimated Image
L	Length of Blur
MAX_I	Highest Possible Signal Intensity
N	Noise Signal
P	Specific Probability

$R(\cdot)$	Regularizer that encourages the Restored Image to be Smooth
W	Window Size
a	Specific Intensity Value
b	Specific Intensity Value
c	Normalization Constant
f	Function of Image
g	Gray Level
i	Iteration
r	Radius of Blur
x	Restored Image
y	Degraded Image
z	Ground Truth
\mathcal{A}	Degradation Operator
α	Threshold Value
β	Momentum Decay Rate
θ	Angle of Blur
λ	Regularization Parameter
μ	Average of Specific Window
σ	Covariance of Specific Window
ϕ	Feature Map
AWGN	Additive White Gaussian Noise
BM3D	Block-Matching and Three-Dimensional Filtering
BN	Batch Normalization
CMY	Cyan-Magenta-Yellow
CNN	Convolutional Neural Network
COC	Circle of Confusion
CPU	Central Processing Unit
CT	Computed Tomography
CUDA	Compute Unified Device Architecture
DIP	Deep Image Prior
DL	Deep Learning
DRCN	Deep-Recursive Convolutional Network

DWT	Discrete Wavelet Transform
ES-EMV	Early Stopping using Exponential Moving Variance
ESPCN	Efficient Sub-Pixel Convolutional Neural Network
FDDNet	Fast and Flexible Denoising CNN
FHT	Fast Hartley Transform
FPN	Feature Pyramid Network
GAN	Generative Adversarial Network
GPU	Graphic Processing Unit
GT	Ground Truth
HR	High-Resolution
LR	Low-Resolution
L-Rate	Learning Rate
L-Rich	Lucy-Richardson
MAE	Mean Absolute Error
MB-ES	Metric-based Early Stopping
OS	Operating System
P	Patience Number
PDF	Point Density Function
PSF	Point Spread Function
PSNR	Peak-Signal-to-Noise Ratio
RAM	Random-Access Memory
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue
RMSE	Root Mean Square Error
RMSE	Root Mean Square Error
SAR	Synthetic Aperture Radar
SF	Scaling Factor
SGD	Stochastic Gradient Descent
SISR	Single Image Super-Resolution
SP	Stopping Point
SRCNN	Super-Resolution Convolutional Neural Network
SSIM	Structural Similarity Index
VDSR	Very Deep Super-Resolution
ZSSR	Zero-Shot Super-Resolution

LIST OF APPENDICES

Appendix A: Additional Tables	98
Appendix B: Additional Figures	103

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Image restoration can be treated as an inverse problem in image processing to recover a high-quality image from a corrupted image. Image restoration is unavoidable as the images are often degraded during the data acquisition process. Image degradation may result in blurring, information loss due to sampling, quantization effects, and numerous types of noise. As its name implies, image restoration attempts to reconstruct an image from its deteriorated original data. It has a wide range of applications, including astronomical imaging, remote sensing, medical imaging, microscope imaging, photographic deblurring and others (Khare et al., 2011).

Digital image processing is superior to analogue image processing, where it offers a greater variety of algorithms that can be applied to the image data, thereby mitigate issues like noise and distortion. There are numerous approaches to image restoration, the two most common of which are filter-based and deep learning-based. Nevertheless, each approach comes with its strengths as well as drawbacks. In order to achieve a higher level of performance, the architecture design for both approaches have been made extremely complicated by integrating a variety of sub-modules. (Mahony et al., 2019). Although the performance has improved, the cost has also increased accordingly. More recently, a novel method known as a deep image prior with the use of handcrafted image priors has emerged (Ulyanov et al., 2018). This approach does not involve any learning and still delivers outstanding performance.

1.2 Importance of the Study

Image quality can be improved from either hardware or software perspective. Although upgrading of hardware specification is a possible option to get higher quality images, it also comes with additional cost. Moreover, there are unavoidable hardware limitations in practise such as sensor imperfection, sensor noises or even faulty sensor. In addition, there are external factors that must be

taken into consideration such as underwater distortion, low light environment, and motion blur.

Recently, many researchers have focused on the development from the software perspective such as super-resolution, image deblurring, and image denoising. They can mitigate the effects caused by the aforementioned factors without overhead hardware cost and especially crucial when various factors are considered.

Hence, this study focuses on improving image restoration without incurring any upgrades and additional expenses on hardware. In general, image restoration is one of the most important steps in image pre-processing. It is critical to enhance the image data by reducing artifacts or enhancing features that will be useful for further analysis and post-processing (Great Learning Team, 2020). Besides, this study is hoped to benefit applications that require accurate data for model training to gain better performance.

1.3 Problem Statement

Problem statements for the current study of image restoration are summarized as follows:

- (i) In real life, the image acquired will be always degraded due to the limitation and flaws in the imaging and capturing process. For example, image degradation due to non-linear and space-variant factors such as random noise signals, underwater photography, camera misfocus, low-light environment and others (Boyat and Joshi, 2015; Lu et al., 2017; Jiang, 2006).
- (ii) Conventional image restoration methods require a careful selection of relevant features from each individual image before it can be applied to the specific image restoration process. This becomes more complicated when greater number of features are involved. In addition, the researcher must work with numerous variables in order to fine-tune each feature definition (Mahony et al., 2019). Therefore, developing a high-quality image restoration model will be a long and cumbersome process.
- (iii) Deep learning (DL) has become popular in various applications involving image processing and analysis. However, DL models

rely heavily on the provided training datasets and neural network architecture. The performance could be significantly downgraded if the architecture of the neural network is poorly designed. Besides that, training a DL model can also take a considerable amount of time as it depends on the processing speed of the hardware resources (Mahony et al., 2019). Hence, the architecture structure of the neural network is required to be properly designed to enhance training speed while retaining performance.

- (iv) Deep image prior (DIP), a learning-free approach to image restoration that proposes using an untrained convolutional neural network with random initialization, has recently proposed. This method takes in a corrupted image and uses a prior derived from the network's parameters, iteratively produces a restored image as the output (Ulyanov et al., 2018). However, this approach requires fine-tuning of hyperparameters and stopping criteria for various image restoration tasks to obtain the desired results.

1.4 Aim and Objectives

This project aims to improve image quality through image restoration techniques. The detailed objectives of this project are listed as follows:

- (i) To provide a comprehensive study on image restoration, in terms of tasks and techniques involved.
- (ii) To investigate the underlying issues of image restoration tasks using DIP.
- (iii) To propose a suitable image restoration method solution in DIP to address the issue identified.

1.5 Scope and Limitation of the Study

This project focuses on the study of image restoration tasks and the methods. State-of-the-art techniques for image restoration are reviewed. In addition, this study further investigates the feasibility of the image restoration approaches and evaluates their application to various types of degradation models.

Within the scope of this project, the performance for image restoration method is evaluated based on selected datasets, which might differ from the original resources. The investigation covers limited datasets and test images but in numbers that are adequate for the study. Besides that, the project only considers some of the noise models that are commonly encountered in real world, given the large variety of noise types that exist.

1.6 Contribution of the Study

This study focuses on the use of DIP in image restoration, which is a relatively new approach compared to classical and deep learning techniques. One of the main benefits of DIP is that it does not require large datasets for machine learning. However, DIP does have its limitations, particularly in fine-tuning parameters. The most critical parameter is early stopping to obtain the best quality image at a specific iteration during the process. This study aims to contribute to the research on DIP by highlighting its benefits and limitations. To address the effectiveness in determining the best image during the DIP restoration process, Metric-based Early Stopping (MB-ES) has been proposed in this study. It is hoped to benefit other researchers who are working with the DIP approach in image restoration.

1.7 Outline of the Report

This report comprises five chapters. Chapter 1 provides an overview of the study, including its importance, problem statement, aim and objectives, scope and limitations, and contributions. Chapter 2 presents a literature review of the related works. Chapter 3 explains the methodology, work plan, and the proposed early stopping detection method to improve the image restoration performance using DIP model. Chapter 4 discusses the results and the comparison with other models. Lastly, Chapter 5 concludes the study and provides recommendations for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In general, the images we see in the real world are continuous signals. Therefore, the analogue information of the visuals is sampled digitally to record and store the image efficiently. An image is a collection of pixel arrays arranged in columns and rows. For instance, a colour image with dimensions of $3 \times 320 \times 640$ will have a total of 614 400 pixels. These pixels represent the signal intensity for each point inside the image. The higher the pixel intensity, the brighter the colour will be observed at that particular location, and vice versa. This section first provides background on digital images and the degradation of image quality due to the wide variety of phenomena. To understand the presence of artifacts in images, blur and noise models are reviewed, as illustrated in Figure 2.1. Accordingly, some image restoration methods are explored and discussed, as outlined in Figure 2.2.

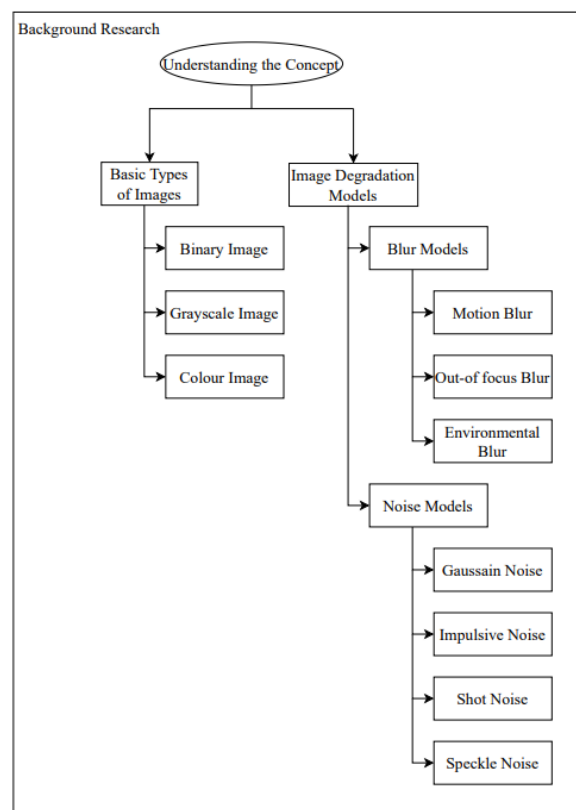


Figure 2.1: Overview of Images Types and Image Degradation Models.

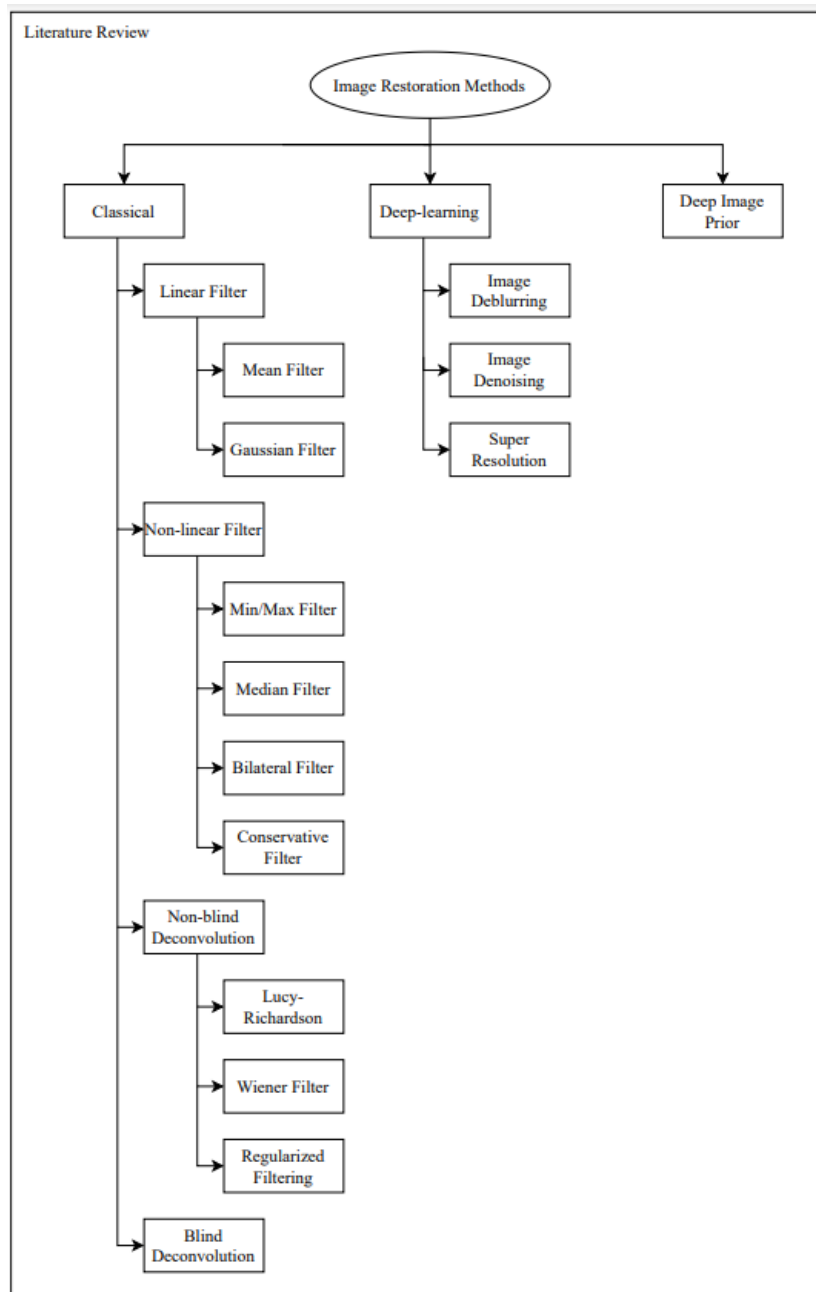


Figure 2.2: Overview of Image Restoration Methods.

2.2 Basic Types of Images

There are basically three types of images which are binary, grayscale, and colour images as shown in Figure 2.3. The following subsections provide further details.

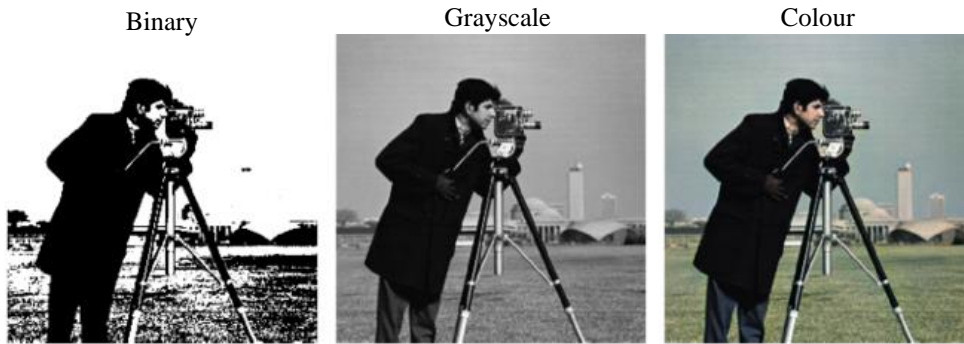


Figure 2.3: Sample of Image Models.

2.2.1 Binary Images

Binary images are images that have been reduced to only two-pixel values which are 0 and 255, representing black and white, respectively. They are widely utilised in applications where the silhouette of an object contains all the necessary information, such as text interpretation and object orientation identification. This binary form of data is essential because the output of edge detection in digital image processing might hold a binary form. Typically, a grayscale image can be thresholded to create binary images using Equation (2.1),

$$f(x, y) = \begin{cases} 255 & \text{for } g > \alpha \\ 0 & \text{for } g \leq \alpha \end{cases} \quad (2.1)$$

where

$f(x, y)$ is the function of image

g = gray level

α = threshold value

2.2.2 Grayscale Images

In general, grayscale images are stored using 8-bit integers, which gives rise to a total of 256 possible different shades of grey. The range of grey shades varies between pure black (0) to pure white (255). Grayscale is commonly used nowadays due to its simple algorithm and lower computational demands. In contrast, colour images require a more complex representation of pixel arrays, which increases the amount of training data and processing speed needed.

2.2.3 Colour Images

Colour images are usually composed of three bands of monochrome arrays each representing a different colour. For example, the red-green-blue (RGB) colour scheme resembles how the retina's receptors perceive colour. RGB model utilizes additive colour mixing and is the fundamental colour model that is used in web graphics, television, computers and others. For printing and filters purposes, the cyan-magenta-yellow (CMY) colour scheme is used instead of RGB. The function of RGB models can be expressed in Equation (2.2),

$$f(x, y) = \begin{bmatrix} r(x, y) \\ b(x, y) \\ g(x, y) \end{bmatrix} \quad (2.2)$$

where

$r(x, y)$ = red channel

$b(x, y)$ = blue channel

$g(x, y)$ = green channel

2.3 Image Degradation Models

2.3.1 Blur Models

A blurring image is obtained after the localized averaging of pixels, that leads to a loss in image sharpness. Common culprits for this blurring effect include lens defocus, changes in the refractive indices of photographic images, or relative motion between the camera and the object being captured. Typically, blurring is modelled by convolving an image with a point spread function (PSF), which is a kernel of a specific size containing data for the convolution process. PSFs are categorized as either spatially invariant or spatially variable. In the former case, the PSF remains the same for all image pixels, while in the latter, the PSF varies for different image pixels. Figure 2.4 illustrates some blurring models including motion blur, out-of-focus blur, and environmental blur.

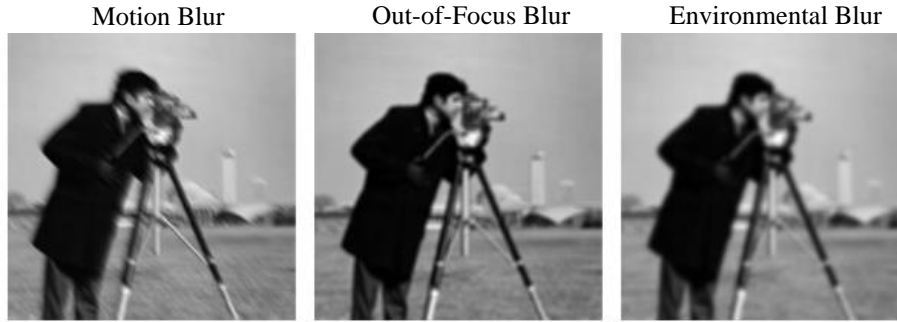


Figure 2.4: Sample of Blurring Models.

2.3.1.1 Motion Blur

Motion blur is a result of the relative movement between an object and an image acquisition device. The movement path must be predicted in order to restore a motion-blurred image. For example, the PSF can be a blur kernel that resembles the movement of the camera during the image capture. The blur is spatially invariant if the camera is moved with respect to the stationary scene. Hence, the PSF for motion blur can be modelled using Equation (2.3) (Kitchener, 2012),

$$PSF(x, y; L, \theta) = \begin{cases} \frac{1}{L} & , \text{if } \sqrt{x^2 + y^2} \leq \frac{L}{2} \text{ for } \frac{x}{y} = \tan \theta \\ 0 & , \text{elsewhere} \end{cases} \quad (2.3)$$

where

L = length of blur

θ = angle of blur

2.3.1.2 Out-of-focus Blur

Typically, the out-of-focus blur appears in the form of a circular disc which is also known as a circle of confusion (COC). The COC diameter depends on factors such as the aperture number, focal length, and the distance between the lens and the object being captured. The lens acts as a low pass filter that eliminates high-frequency spectrum when the image acquisition device is out-of-focus. The PSF for out-of-focus blur is modelled using Equation (2.4) (Kitchener, 2012),

$$PSF(x, y; r) = \begin{cases} \frac{1}{\pi r^2}, & \text{if } \sqrt{x^2 + y^2} \leq r \\ 0, & \text{elsewhere} \end{cases} \quad (2.4)$$

where

r = radius of blur

2.3.1.3 Environmental Blur

Natural phenomena such as the bending and scattering of light through materials with different refractive indices can lead to a degradation of image quality. For instance, the underwater images are distorted due to wavelength absorption which reduces the colour intensity. On top of that, there is a strong correlation between the probability of wavelength absorption and the salinity of the water. The PSF of environmental blur can be derived from Gaussian distribution, as shown in Equation (2.5) (Kitchener, 2012),

$$PSF(x, y; \sigma_h^2) = c \exp\left(-\frac{x^2 + y^2}{2\sigma_h^2}\right) \quad (2.5)$$

where

σ_h^2 = variance of blur

c = normalization constant

2.3.2 Noise Models

Image noise can be defined as random fluctuation in pixel intensity that degrades the original image by adding irrelevant and meaningless content in terms of colour or brightness information. The extent of this degradation depends primarily on the frequency distribution of the noise source. Figure 2.5 shows the common noises which are Gaussian, Impulsive/Salt-and-Pepper, Shot/Poisson, and Speckle noises. The model of a noisy image can be expressed in Equation (2.6),

$$A = B + N \quad (2.6)$$

where

A = Noisy image

B = Original image

N = Noise signal

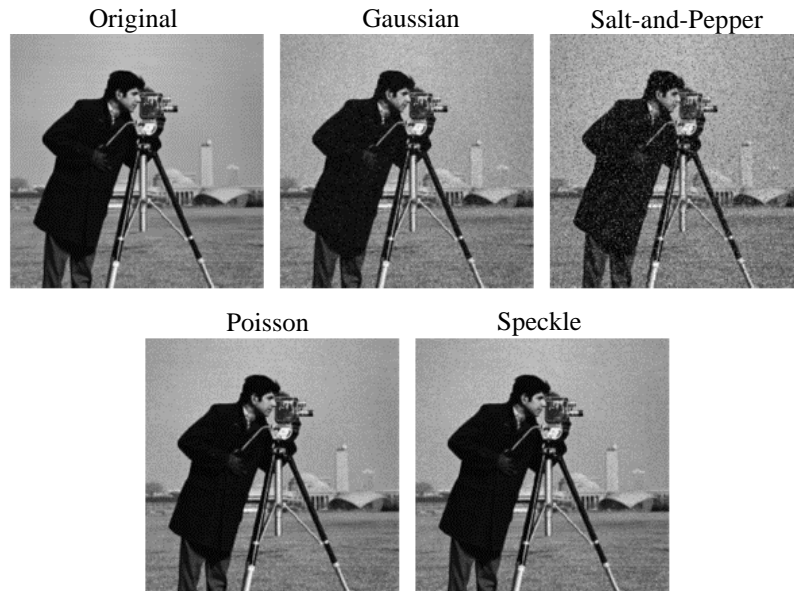


Figure 2.5: Sample of Different Types of Noise Models.

2.3.2.1 Gaussian Noise

Gaussian noise is a noise that statistically possesses a bell-shaped probability density function, which is similar to a normal distribution as shown in Figure 2.6. Hence, the noisy image is made up where each pixel is the summation of the original pixel with a noise value that followed Gaussian distribution. One of the most common applications is additive white Gaussian noise (AWGN). The probability density function (PDF) of Gaussian distribution is shown in Equation (2.7) (Rani et al., 2016),

$$PDF_{Gaussian} = \frac{1}{\sqrt{2\pi}\sigma_N} e^{-\frac{(g-m_f)^2}{2\sigma_N^2}} \quad (2.7)$$

where

σ_N = standard deviation of noise

g = grey level measurement

m_f = mean of the function

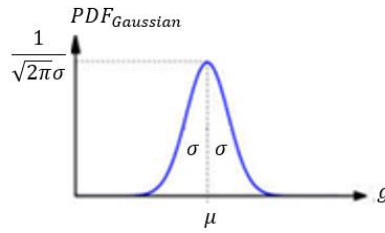


Figure 2.6: Graphical Representation of PDF for Gaussian Noise (Source: Kanrar and Maji, 2022).

2.3.2.2 Impulsive Noise

Impulse noise, also referred to as salt-and-pepper noise, is characterized by the presence of white and black pixels that are randomly distributed throughout the image. On average, the noise affects less than 0.1% of the pixels. Impulse noise is typically caused by errors in data transfer and can have a very high intensity. The corrupted pixels are either set to 0 (pepper noise) or 255 (salt noise), while the rest of the pixels remain unaffected. Impulse noise is commonly found in faulty memory locations, errors in an analogue-to-digital converter, malfunctioning camera sensors, and other similar situations. The PDF of salt-and-pepper noise is expressed in Equation (2.8) (Rani et al., 2016),

$$PDF_{salt-and-pepper} = \begin{cases} P_{salt} & \text{for } g = \text{salt} \\ P_{pepper} & \text{for } g = \text{pepper} \end{cases} \quad (2.8)$$

where

P_{salt} = the probability of salt granule

P_{pepper} = the probability of pepper granule

g = grey level measurement

2.3.2.3 Shot Noise

Shot noise or Poisson noise which is obtained from electromagnetic sources such as visible lights, x-rays, gamma rays and others. It is a fundamental type of uncertainty related to light measurement due to the independence of photon detections and the quantized nature of light. The fluctuation of photons emitted

by these sources results in an image that is spatially and temporally unpredictable.

2.3.2.4 Speckle Noise

In digital and optical computer vision, speckle noise is one of the fundamental issues during the image restoration process. Unlike Gaussian and salt-and-pepper noises, it is multiplicative which can be obtained by multiplying randomized pixel values with various image pixels. The image quality is diminished by the granular interference of speckle noise. Such noise mostly occurs in Synthetic Aperture Radar (SAR), medical images, lasers and others. The PDF of speckle noise is followed by a gamma distribution (see Figure 2.7) which is expressed in Equation (2.9) (Kanrar and Maji, 2022),

$$PDF_{Speckle} = \frac{a^b g^{(b-1)}}{(b-1)!} e^{-ag} \quad (2.9)$$

where

a and b = intensity values

g = grey level measurement

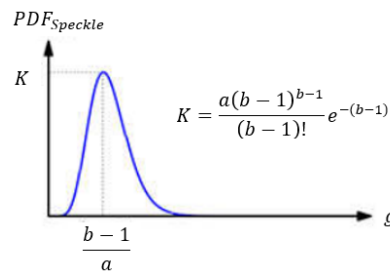


Figure 2.7: Graphical Representation of PDF for Speckle Noise (Source: Kanrar and Maji, 2022).

2.4 Classical Image Restoration Methods

2.4.1 Linear Filter

Generally, linear filtering can be used to remove noise from an image. There are particular filters that are suitable for this task, such as the averaging filter and the Gaussian filter. For instance, an averaging filter can be used to eliminate

grain noise in an image. Local fluctuations caused by grain are eliminated by averaging each pixel's neighbours. In fact, linear filtering is defined as a process at which the value of an output pixel is calculated by adding together the values of all the neighbouring input pixels linearly. This is achieved through convolution, where each output pixel is computed as a weighted sum of adjacent input pixels. A kernel or filter matrix is used during the convolution process to perform the linear operation. One major drawback of the convolution filter is that it is not effective in dealing with all types of noise. Figure 2.8 shows sample outputs with different types of linear filters.

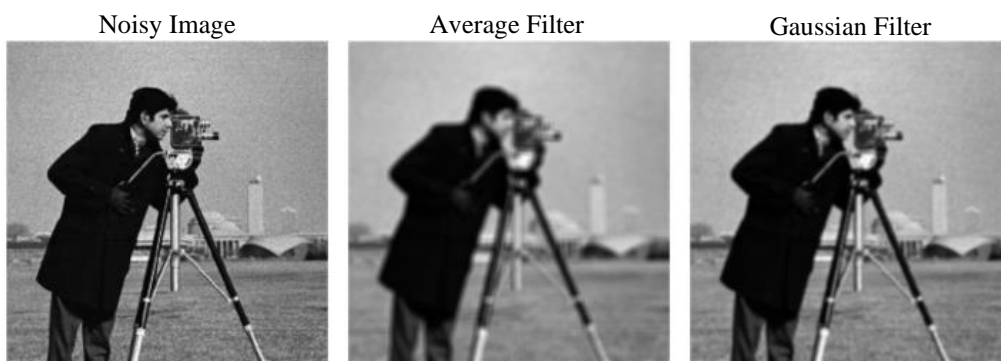


Figure 2.8: Denoising Outputs Using Linear Filters.

2.4.1.1 Average Filter

The fundamental mean filter is a simple noise reduction technique that replaces the value of each pixel in an image with the average value of the pixels around it, including itself. This can be accomplished using a convolution kernel with coefficients that all have the same value. The mean filter computes the average value of the neighbouring pixels for each central pixel in the image, resulting in noise reduction. Figure 2.9 illustrates a convolution process between a matrix with a 3×3 square kernel matrix.

Matrix with random pixels value		Kernel 3 x 3 Pixels		Output matrix after convolution process																																																											
<table style="border-collapse: collapse; text-align: center;"> <tr><td>4</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td style="background-color: #e0f0ff;">7</td><td>8</td><td>7</td><td>4</td></tr> <tr><td>6</td><td>5</td><td>6</td><td>5</td><td>6</td></tr> <tr><td>2</td><td>3</td><td>9</td><td>7</td><td>8</td></tr> <tr><td>7</td><td>1</td><td>4</td><td>3</td><td>2</td></tr> </table>	4	2	3	4	5	1	7	8	7	4	6	5	6	5	6	2	3	9	7	8	7	1	4	3	2	x	<table style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	1	1	1	1	1	=	<table style="border-collapse: collapse; text-align: center;"> <tr><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td style="background-color: #e0f0ff;">4.67</td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> </table>							4.67																		
4	2	3	4	5																																																											
1	7	8	7	4																																																											
6	5	6	5	6																																																											
2	3	9	7	8																																																											
7	1	4	3	2																																																											
1	1	1																																																													
1	1	1																																																													
1	1	1																																																													
	4.67																																																														
		x $\frac{1}{9}$																																																													
		Average Filter																																																													

$$4.67 = \frac{1}{9} [4(1) + 2(1) + 3(1) + 1(1) + 7(1) + 8(1) + 6(1) + 5(1) + 6(1)]$$

Figure 2.9: Convolution Process of Average Filter.

From Figure 2.8, it can be observed that the output image appears smoother, but the edges of the pixels have been affected. The presence of a single outlier pixel can significantly alter the average value of its neighbouring pixels. When the filter's neighbourhood spans an edge, it interpolates new values for border pixels, resulting in a substantial softening of the boundary. This could be problematic if the desired output needs to have sharp edges.

2.4.1.2 Gaussian Filter

Gaussian smoothing is a method of image processing that uses a two-dimensional (2D) convolution operator to blur the image and get rid of the noise details. It shares some similarities with the mean filter, but the only difference is due to the kernel applied, which is an approximation of Gaussian distribution. During the convolution process, the center of the filter receives the most weight, while the significance decreases as it moves away from the center. Figure 2.10 demonstrates the convolution process between a matrix with a 3×3 Gaussian kernel.

Matrix with random pixels value		Kernel 3 x 3 Pixels		Output matrix after convolution process																																																											
<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>4</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>7</td><td>8</td><td>7</td><td>4</td></tr> <tr><td>6</td><td>5</td><td>6</td><td>5</td><td>6</td></tr> <tr><td>2</td><td>3</td><td>9</td><td>7</td><td>8</td></tr> <tr><td>7</td><td>1</td><td>4</td><td>3</td><td>2</td></tr> </table>	4	2	3	4	5	1	7	8	7	4	6	5	6	5	6	2	3	9	7	8	7	1	4	3	2	x	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>3</td><td>5</td><td>3</td></tr> <tr><td>5</td><td>9</td><td>5</td></tr> <tr><td>3</td><td>5</td><td>3</td></tr> </table> <p style="text-align: center; margin: 0;">Gaussian Filter</p>	3	5	3	5	9	5	3	5	3	=	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td>4.88</td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> </table>							4.88																		
4	2	3	4	5																																																											
1	7	8	7	4																																																											
6	5	6	5	6																																																											
2	3	9	7	8																																																											
7	1	4	3	2																																																											
3	5	3																																																													
5	9	5																																																													
3	5	3																																																													
	4.88																																																														
$4.88 = \frac{1}{41} [4(3) + 2(5) + 3(3) + 1(5) + 7(9) + 8(5) + 6(3) + 5(5) + 6(3)]$																																																															

Figure 2.10: Convolution Process of Gaussian Filter.

The Gaussian filter is able to produce an image with more details as compared to the uniform weights of an average filter. It is due to the characteristic of Gaussian distribution, where the average weights more heavily focused on the values of the central pixels. Hence, given a similar size of the kernel, the Gaussian filter outperforms the average filter by smoothing the image more moderately while preserving edges.

2.4.2 Non-linear Filter

In the previous section, all of the linear methods listed above yield the same result: the image's structure, lines, and edges are blurred. This is because a linear filter is unable to eliminate noise caused by a single pixel with a high spike intensity that can significantly impact the weighted average of a kernel. As a result, it is inevitable to strike a balance between removing noise and preserving image details. Hence, non-linear filters are developed to rectify this issue by being able to get rid of any single outlier intensity values, such as the maximum or minimum filter, conservative filter, median filter, bilateral blurring filter and others.

2.4.2.1 Maximum Filter and Minimum Filter

The minimum and maximum filters give the minimum and maximum values in a moving region of the original image. The former replaces the central pixel with the darkest value among its neighbouring pixels, while the latter replaces it with the lightest value. Figure 2.11 shows the denoising outputs from both filters in removing the salt-and-pepper noise, respectively.



Figure 2.11: Denoising Outputs Using Minimum and Maximum Filters.

It can be observed that the edges have been successfully preserved, but the denoising performance is not optimal as some noise is still present in both cases.

2.4.2.2 Median Filter

The median filter is a popular image noise reduction technique, similar to the mean filter, but with the added benefit of preserving more important image

details. Unlike the mean filter, it uses the median value of neighbouring pixels instead of their average to replace the pixel's original value. This algorithm sorts the pixel values in numerical order to compute the median value, which is then used to replace the central pixel. Figure 2.12 demonstrates the calculation of the median value from a pixel neighbourhood.

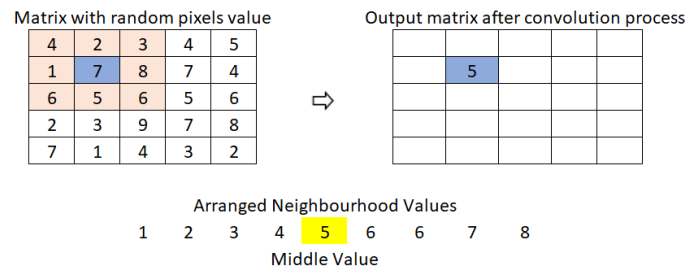


Figure 2.12: Calculation of Median Value with 3×3 Kernel.

Figure 2.13 depicts the denoising output of a median filter applied to remove salt-and-pepper noise. It is noticeable that the noise granules have been effectively removed from the degraded image. This is because the median filter replaces the central pixel value with the median value of the surrounding pixels, which preserves the original information. Hence, the impact of salt-and-pepper noise on the median filter is less significant than on the average value, as the median is less likely to be skewed by outliers. As a result, the median filter does not produce artificial pixel values when it spans an edge because the median value always corresponds to the value of any of the neighbouring pixels. Therefore, this is the reason the median filter is far superior to the mean filter in maintaining the sharpness of edges.

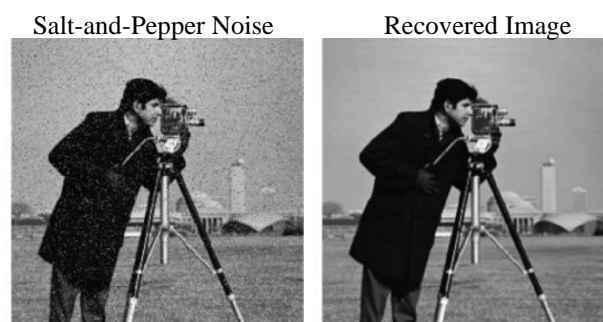


Figure 2.13: Denoising Output Using Median Filter to Remove Salt-and-Pepper Noise.

Nevertheless, the median filter is difficult to calculate and can be rather costly since the sorting of values into numerical order is considerably time-consuming and slow. Hence, a smart algorithm was developed to enhance the efficiency by taking advantage of the reappearance of the neighbouring pixels when the window is slide over an image (Zhu and Huang, 2012).

2.4.2.3 Conservative Filter

The conservative filter is an image noise reduction technique that uses a fast and simple filtering algorithm to preserve high spatial frequency details in an image. However, it is less effective in removing additive noise, such as Gaussian noise, as it is favourably towards removing clusters of pixels or noise spikes with unusually low or high pixel intensity, like salt-and-pepper noise. Typically, noise is assumed to have a high spatial frequency and can be reduced by making each pixel's intensity comparable with its neighbouring pixels. Unlike average and median filtering, conservative smoothing ensures that each pixel's intensity is within its neighbours' range. The overall process of a conservative filter can be described in the following procedures (Fisher et al., 2003):

- (i) Minimum and maximum pixel intensity values are determined by considering all the surrounding pixels within a windowed region.
- (ii) The central pixel of the image is carried over to the output image without any modifications if its intensity falls within the intensity ranges of neighbouring pixels.
- (iii) If the intensity of the central pixel is greater than the maximum value of its surrounding pixels, it will be replaced with the maximum value, and vice versa.

Figure 2.14 depicts the denoising outputs using conservative filter to remove salt-and-pepper noise at different levels. Although it does not eliminate as much noise as the median filter, it is able to preserve more details such as edge sharpness. However, this method is only suitable for cases with low levels of salt and pepper noise, as demonstrated in Figure 2.14. It is less effective when the image is heavily corrupted, with multiple pixels in the neighbourhood being affected.

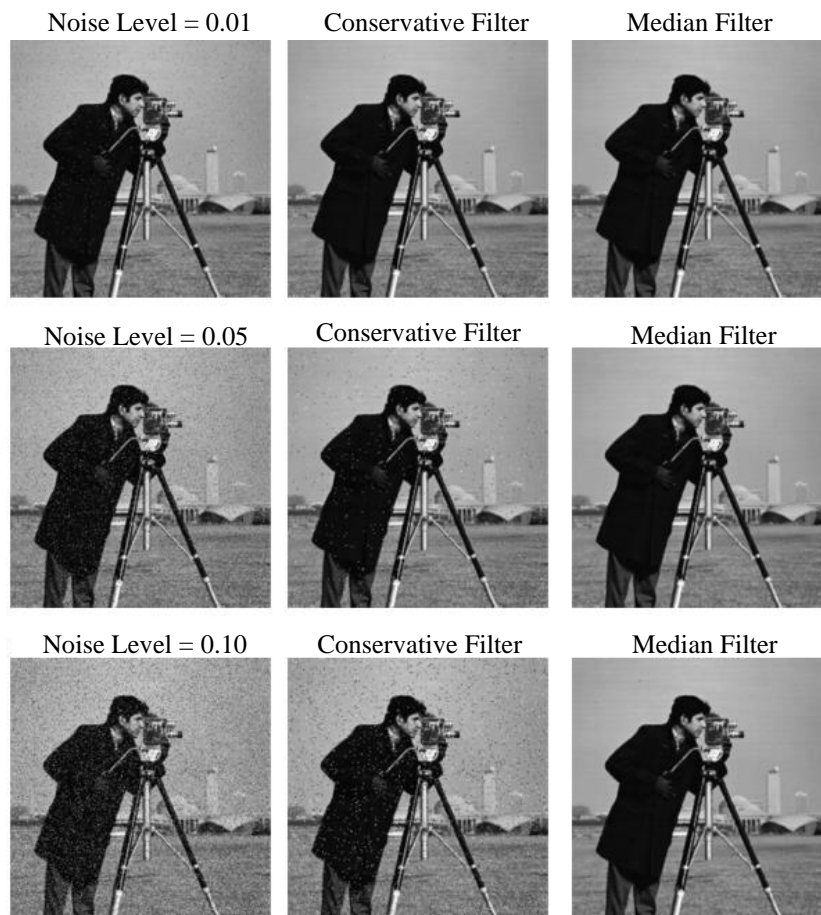


Figure 2.14: Comparison between the Performance of Conservative Filter and Median Filter due to Different Salt-and-Pepper Noise Levels.

2.4.2.4 Bilateral Blurring Filter

Blurring images to minimize noise and details has a side effect of losing image edges. Hence, bilateral blurring is utilized to reduce noise while preserving edges. The algorithm of bilateral blurring includes two Gaussian distributions. The first Gaussian function considers only spatial neighbours, while the second considers neighbouring pixel intensity, ensuring that only related pixels are included in the blur computation. If nearby pixels within the same neighbourhood possess similar values, it is reasonable to assume they are both referring to the same entity. However, if they have different values, it can be inferred that they are referred to the boundary or edge of an object. Compared to the aforementioned methods, the main drawback of this filter is its

considerably slow processing speed. Figure 2.15 shows the result of removing Gaussian noise with a bilateral filter.

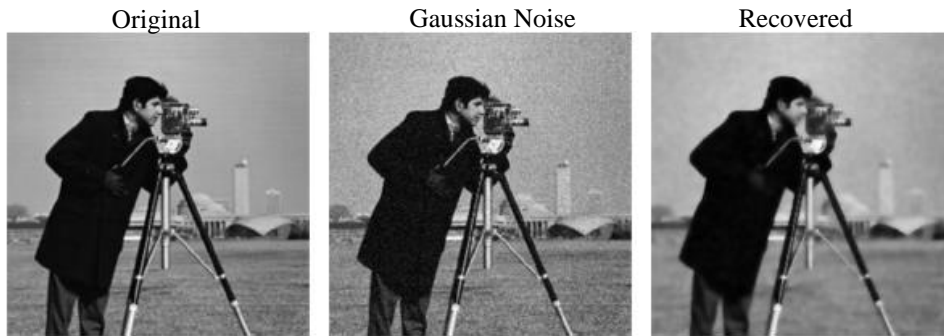


Figure 2.15: Removal of Gaussian Noise with Bilateral Filter.

2.4.3 Non-blind Deconvolution for Image Restoration

This type of image restoration method requires prior knowledge of the parameters of PSF such as its angle and length. In this section, Lucy-Richardson (L-Rich), Wiener filter and regularized filtering deconvolution methods will be discussed.

2.4.3.1 Lucy-Richardson (L-Rich) Deconvolution

L-Rich Deconvolution is a non-blind method that is used to restore a degraded image with a known PSF. In the early 1970s, the L-Rich deconvolution method gained significant attention in the fields of astronomy and medical imaging. This method was initially derived using Bayes' theorem (Richardson, 1972). The parameters of the reconstructed image are modelled with the use of Bayesian statistics probability distribution to image restoration methods. The latent image and the PSF are both restored simultaneously by the algorithm through an iterative process, where the pixels are given by,

$$d_i = \sum p_{ij}u_j \quad (2.10)$$

where

d_i = observed value at the position- i

p_{ij} = PSF from the true location of position- j relative to the observation position- i

u_j = latent image at position- j

The iterative equation for estimating the most likely restored image model is obtained by rearranging Equation (2.10), resulting in Equation (2.11):

$$u_j^{(t+1)} = u_j^{(t)} \sum_i \frac{d_i}{c_i} p_{ij} \quad (2.11)$$

where

$$c_i = \sum_j p_{ij} u_j^{(t)}$$

However, according to the study, they claimed that the convergence of the L-Rich iteration is relatively very slow due to the non-linear algorithm. Besides that, the number of iterations must be manually set for each image based on the PSF size in order to produce a restored image of excellent quality. Despite this drawback, this method is widely used because of its ability to achieve maximum likelihood implementation and produce high-quality reconstructions even in the presence of significant noise levels. Figure 2.16 demonstrates the process of blurring an image with a random PSF, while Figure 2.17 shows the deblurring process of the blurred image and noisy blurred image by adopting the L-Rich blind deconvolution method, respectively.

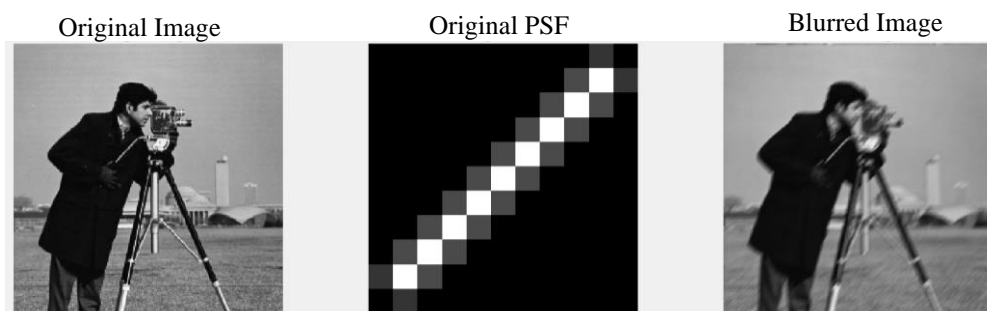


Figure 2.16: Blurring of an Image with a Given Random PSF.

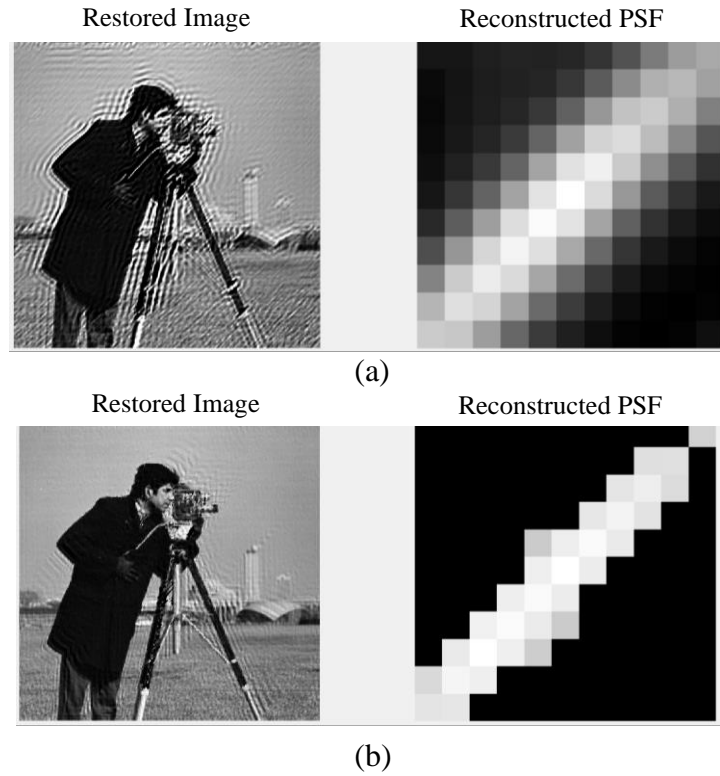


Figure 2.17: Example of Output Using Lucy-Richardson Deconvolution for (a) Blurred Image, and (b) Noisy Blurred Image.

The obtained images from both degradation models have high resolution and better quality, which is attributed to the simplicity of Fourier transformation requiring less computation. However, the main problem with the L-Rich method is the presence of a ringing effect, which can be observed in both images. As a result, a modified L-Rich method was proposed by Sharma et al. (2013), which divides the discrete wavelet transform (DWT) of the degraded image into four sub-frequencies bands, with each being subsequently applied to the L-Rich method.

2.4.3.2 Wiener Filter Deconvolution

Wiener filtering is also a non-blind algorithm used to restore degraded images. It is widely applied in several fields such as echo cancellation, linear prediction, channel equalization, signal restoration and others. The Wiener filter is capable of eliminating both the additive noise and the blurring at the same time, making it a two-in-one process. This is achieved by performing deconvolution through inverse filtering (high-pass filtering) while simultaneously removing noise with

a compression operation (low-pass filtering). The objective of this method is to determine the restoration function that best approximates the original image so that the mean square error can be minimized as much as possible. The deconvolution output can be modelled as Equation (2.12),

$$B' = f_W * (B + N) \quad (2.12)$$

where

B' = estimated image

f_W = function of Wiener Filter

Figure 2.18 shows that the Wiener Filter deconvolution algorithm is able to estimate the PSF effectively to recover the image. The optimal trade-off between inverse filtering and noise smoothing allows the filter to simultaneously remove additive noise and invert the blurring. Another advantage of the Wiener filter is its ability to incorporate the power spectra of the original image and additive noise, without the concern of singularity in inverse filtering.



Figure 2.18: Example of Output from Wiener Filter Deconvolution.

Despite its effectiveness, the Wiener filter has some limitations. The inverse filtering approach is highly susceptible to the presence of additive noise, making it difficult to accurately estimate the power spectra and achieve optimal restoration results. As a solution, a modified noise estimation that proposed by Shimamura et al. (2009) that considers the noise power spectrum in both low

and high-frequency regions. Furthermore, Wiener filters are fairly slow to be used since the operation is always involved in the frequency response domain (Das et al., 2015). Hence, the fast Hartley transform (FHT) had been integrated with the Wiener filter to improve the speed of the deblurring process (Zheng, 1989).

2.4.3.3 Regularized Filter Deconvolution

Regularized filtering is an effective technique when the smoothness of an image is constrained and there is limited information about additive noise. It utilizes a constrained least square restoration algorithm to restore sharpness and remove noise from a noisy and blurred image. Regularized filtering requires less prior knowledge compared to the Wiener filter to apply restoration. This approach can be helpful when no statistical information is provided. Additionally, the regularized filtering framework can be modified to handle image edges, spatially varying noise, and other challenges. Figure 2.19 shows the restoration process of a given blurred and noisy image using regularized filtering deconvolution.

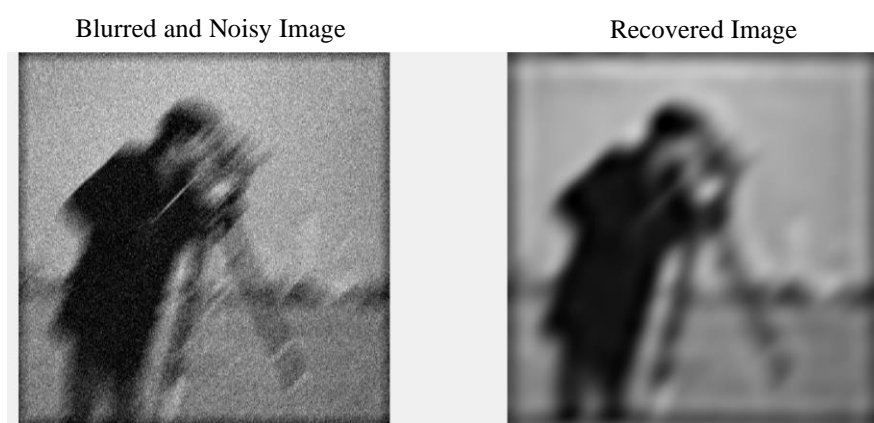


Figure 2.19: Example of Output from Regularized Filtering Deconvolution.

2.4.4 Blind Deconvolution Image Restoration

As its name implies, blind deconvolution is a method of deconvolution that involves no prior information regarding the PSF of a degraded image. This approach enables the target image to be recovered from a single blurred image or a group of blurred images even when the PSF is unknown. It can be done

using either an iterative or non-iterative approach. In the iterative approach, the PSF estimation is updated with each iteration, and the improved PSF is then used to refine the final image until it more closely resembles the original image. In contrast, the non-iterative method applies an algorithm that uses external information to recover the PSF for image restoration (Navaneethakrishnan, 2014).

Besides that, blind deconvolution can be further classified into two types: projection-based and maximum-likelihood methods. The former restores the PSF and true image simultaneously, while the latter estimates the blur parameters, including the PSF and covariance matrices. As a result, the former approach is less sensitive to noise and can easily support size irregularities, while the latter approach provides low computational complexity and facilitates the estimation of blur, noise, and power spectra of the image (Yadav et al., 2016).

Table 2.1 demonstrates that blind deconvolution outperforms other non-blind methods in terms of mean square error (MSE), root mean square error (RMSE) and peak signal-to-noise ratio (PSNR). However, A.M et al. (2014) have identified three major drawbacks to blind deconvolution:

- (i) Accurate knowledge of the PSF is crucial for achieving better performance.
- (ii) Some PSFs do not have frequency zeros, which can lead to inaccurate PSF estimation.
- (iii) The presence of additive noise can mask the frequency-domain nulls and result in performance degradation.

Table 2.1: Evaluation Metrics on Restoration Outputs (Source: Navaneethakrishnan, 2014). Best score is underlined.

	L-Rich	Regularized Filtering	Wiener Filter	Blind Deconvolution
MSE ↓	207.98	1112.71	366.66	<u>138.29</u>
RMSE ↓	10.31	33.96	19.15	<u>11.76</u>
PSNR ↑	24.98	17.70	22.52	<u>26.76</u>

Note: The symbols ↓ and ↑ indicate that lower and higher values, respectively, represent better performance for the corresponding metric.

2.4.5 Investigation of Classical Image Restoration Methods

This section presents a comparison of classical image restoration techniques, including linear and non-linear filters, and deconvolution methods. Examples of the results obtained from these methods will be provided along with explanations.

2.4.5.1 Linear and Non-linear Filters Methods

Figure 2.20 and Figure 2.21 show the denoising results on cameraman image using linear and non-linear filters for salt-and-pepper and Gaussian noises, respectively. The restored images are compared to the ground truth (GT). Table 2.2 shows the performance comparison accordingly.

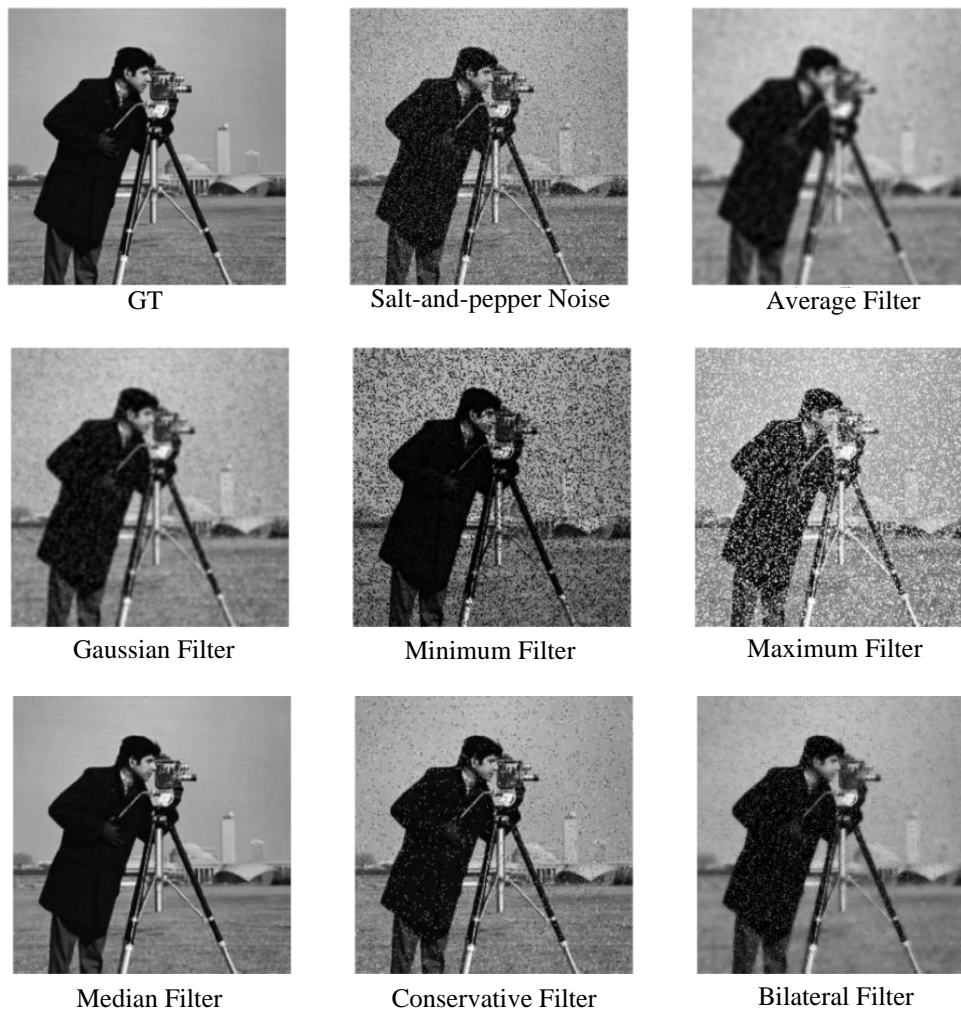


Figure 2.20: Salt-and-Pepper Denoising Results on Cameraman Image Using Classical Methods.

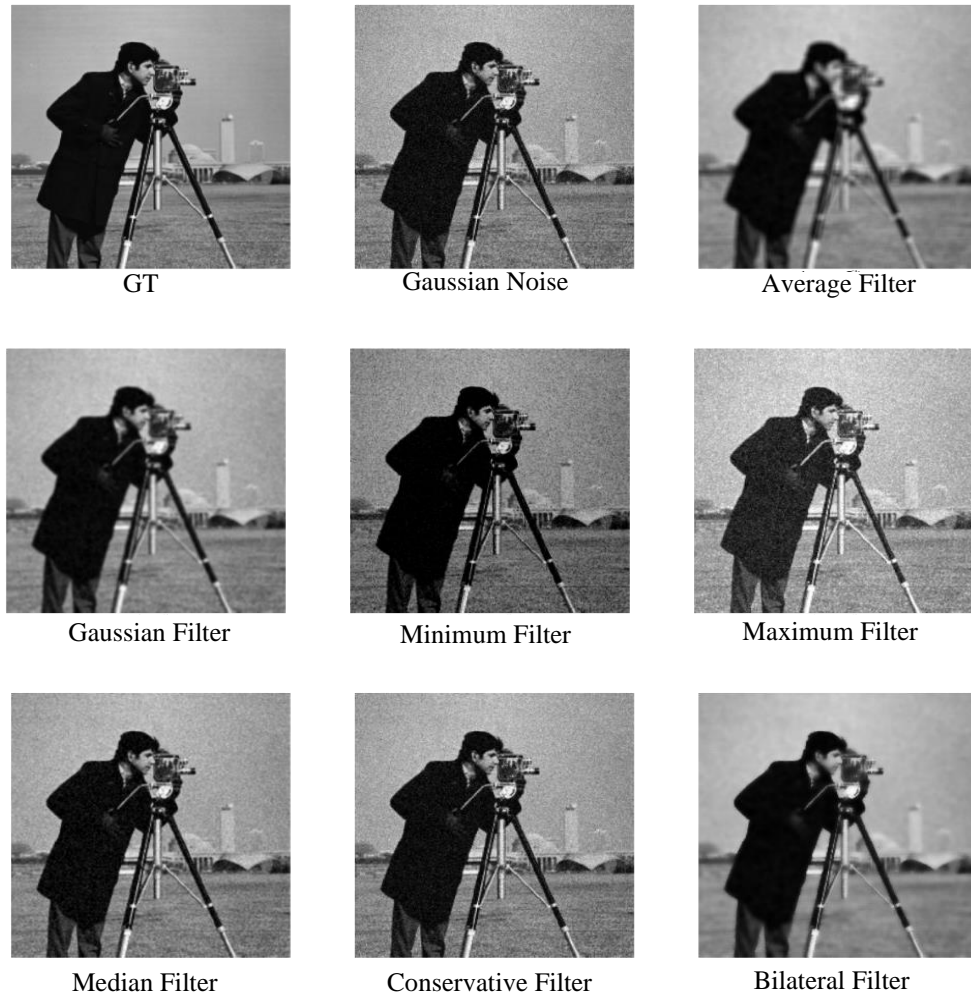


Figure 2.21: Gaussian Denoising Results on Cameraman Image Using Classical Methods.

Table 2.2: Performance Comparison on Cameraman Image Using Classical Methods. Best score is underlined.

Method	Salt-and-Pepper Noise			Gaussian Noise		
	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑
Average Filter	21.2006	21.6038	0.6026	19.8688	22.1674	0.6511
Gaussian Filter	17.0535	23.4945	0.6359	14.7776	24.7387	0.7306
Minimum Filter	59.3693	12.6596	0.2541	33.1294	17.7265	0.3138
Maximum Filter	65.6352	11.7881	0.1457	35.4143	17.1472	0.3064
Median Filter	<u>8.9133</u>	<u>29.1301</u>	<u>0.9184</u>	13.4765	25.5392	0.5315
Conservative Filter	26.4358	19.6870	0.5692	20.3595	21.9555	0.3433
Bilateral Filter	28.3336	19.0848	0.3269	<u>12.5563</u>	<u>26.1536</u>	<u>0.7371</u>

Note: The symbols ↓ and ↑ indicate that lower and higher values, respectively, represent better performance for the corresponding metric.

From Table 2.4, it can be observed that the median filter outperforms the other methods in removing salt-and-pepper noise. In contrast, bilateral filter is effective in reducing Gaussian noise. Therefore, it can be concluded that different algorithms are required to address different types of noise signals.

2.4.5.2 Deconvolution Methods

In this subsection, three methods which are L-Rich, Wiener filter, and regularized filtering techniques were employed for deconvolution. Figure 2.22 and Figure 2.23 display the deconvolution results for blurred, and blurred and noisy images, respectively. A comparison of the two scenarios is presented in Table 2.2.

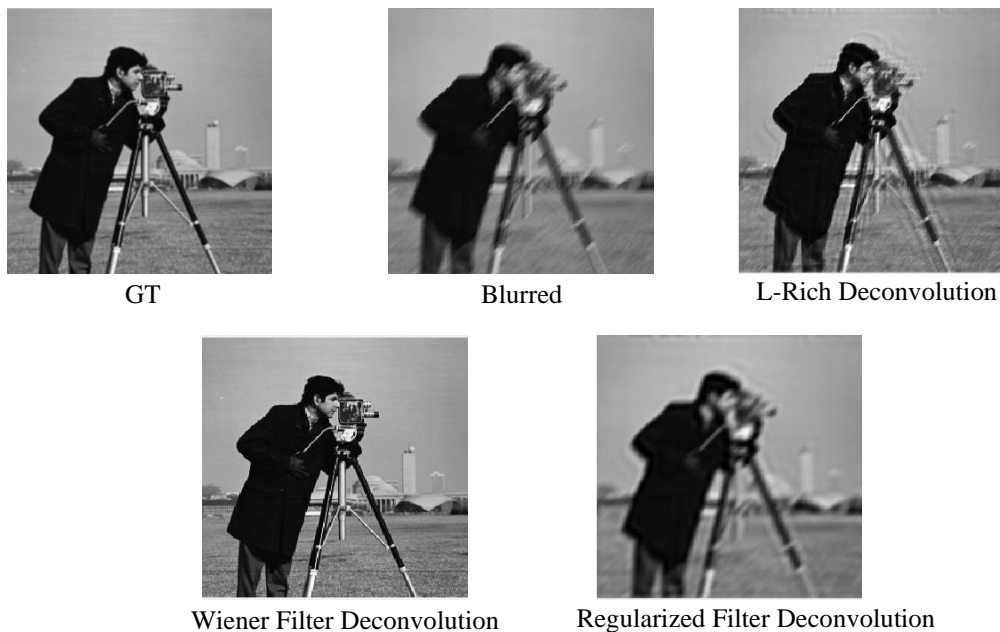


Figure 2.22: Restoration Results on Blurred Cameraman Image Using Deconvolution Methods.

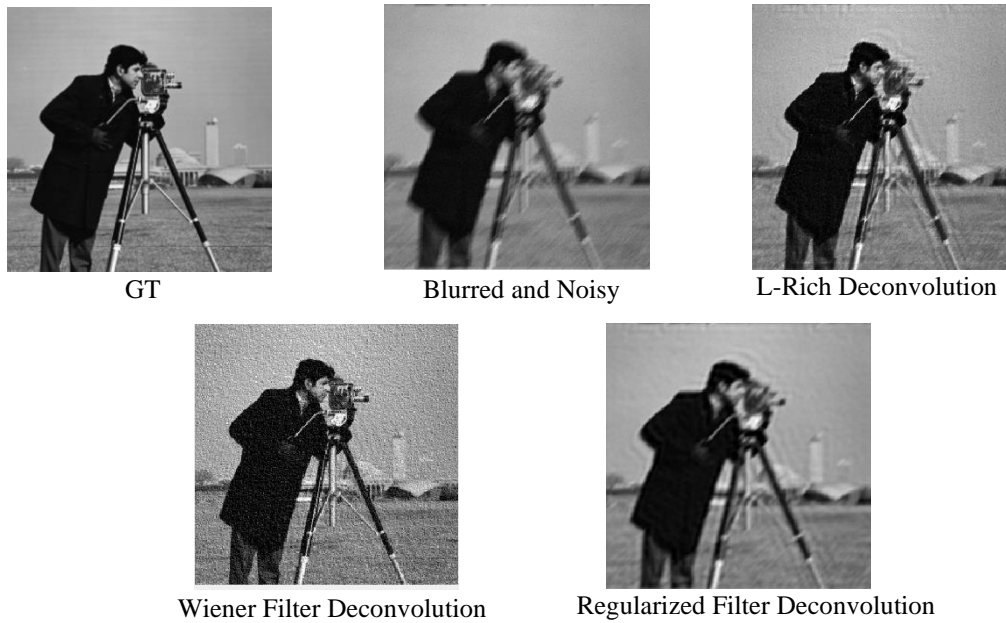


Figure 2.23: Restoration Results on Blurred and Noisy Cameraman Image Using Deconvolution Methods.

Table 2.3: Performance Comparison on Cameraman Image Using Deconvolution Methods. Best score is underlined.

Method	Blurred			Blurred and Noisy		
	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑
Lucy-Richardson	13.3641	25.6120	0.8125	20.7169	21.8043	0.6406
Wiener Filter	<u>0.3025</u>	<u>58.5148</u>	<u>0.9992</u>	22.0034	21.2810	0.3525
Regularized Filter	16.5640	23.7475	0.7425	<u>13.5989</u>	<u>25.4607</u>	<u>0.7298</u>

Note: The symbols ↓ and ↑ indicate that lower and higher values, respectively, represent better performance for the corresponding metric.

Table 2.2 shows that the L-Rich method is able to provide satisfactory results but there are significant ringing effects, as shown in Figure 2.22. The Wiener filter method produces better results for the blurred image but its performance deteriorates with the addition of noise. The regularized filtering technique provides consistent performance for both scenarios because it uses a constrained least square restoration method to handle blurring and noise effects.

However, since the PSF is known for this case, the deconvolution methods are impractical. To make it practical, the model needs to estimate the PSF itself through an iterative process to restore the image.

2.4.6 Summary

Since each image restoration task requires a different algorithm, thus traditional methods are notoriously difficult to build. Hence, this will lead to time-consuming investigations and improvements to the traditional methods. To address these issues, the following section will cover the development of cutting-edge technology known as deep learning (DL) approaches.

2.5 Deep Learning (DL) Approaches to Image Restoration

Over the past decade, breakthroughs in deep learning had a profound effect on various computer vision tasks including recognition, classification and regression. In general, deep learning allows parameters to be learned directly from the available resources without any intervention. In addition, deep learning is not limited to linearity and is capable of learning transformations that are arbitrary complicated and non-linear. As an illustration, a deep learning model is able to develop its own complicated algorithm to restore noisy images to clean ones. Su et al. (2022) reviewed a wide variety of deep learning-based image restoration models, and found that the majority of these models can reach a higher level of performance as compared to state-of-the-art alternatives. In this section, some DL models related to image denoising, image deblurring and super-resolution are briefly discussed.

2.5.1 Image Deblurring

In 2017, Nah et al. proposed a multi-scale deblurring network to restore an image in stages by using a “coarse-to-fine” structure. The multi-scale networks were initially developed by Eigen et al. (2014) that made up of two components: coarse-scale and fine-scale networks. The former is used to predict the depth of an image structure globally by analysing high-level features, while the latter refines the coarse prediction locally by interpreting low-level features. Combining the two networks can fully utilise the distributions of the raw data, resulting in better performance. Figure 2.24 shows the model architecture of a multi-scale structure. Besides that, Zhang et al. (2019) made some modifications to the multi-scale networks by adopting residual learning at different levels to improve the inferencing speed and applying spatial pyramid matching to allow more training data to be refined at the finest level.

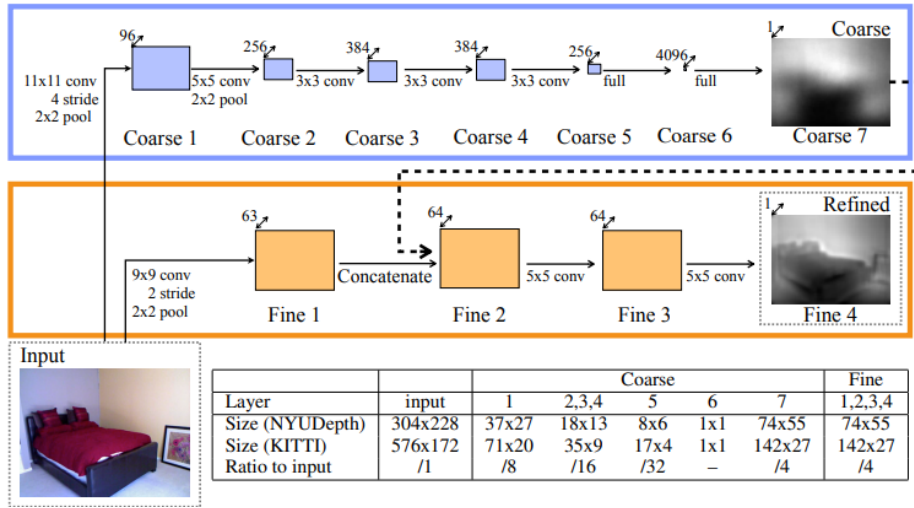


Figure 2.24: Multi-Scale Network. (Source: Nah et al., 2017).

Kupyn et al. (2018) introduced the DeblurGAN model, which is the first use of conditional generative adversarial network (GAN) for deblurring purposes. The key elements of the proposed generator are the residual network block and perceptual loss function, which is given by,

$$L_x = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left[\phi_{i,j}(I^S)_{x,y} - \phi_{i,j}(G_{\theta_G}(I^B))_{x,y} \right]^2 \quad (2.13)$$

where

$\phi_{i,j}$ = feature map after j -th convolution before i -th maximum pooling layer within a network

$W_{i,j}$ = width of the feature maps

$H_{i,j}$ = height of the feature maps

G_{θ_G} = Generator function

I^S = smooth image

I^B = blurred image

The proposed approach encourages the restoration of finer texture details and is capable of finding solutions for those indistinguishable pixels between blurred and sharp images. However, the training speed for the

DeblurGAN network is relatively slow which took about 7 days. Therefore, Kupyn et al. (2019) proposed another version of the deblurring network, DeblurGAN-v2 with a new generator that is constructed based on the Feature Pyramid Network (FPN). FPN generates several feature map layers with improved semantics and information, using a structure that is compromised by bottom-up and top-down approaches. The former uses a convolutional network to collect features, downsampling the spatial resolution while extracting and compressing semantic context; whilst the latter restores semantically rich layers at a higher spatial resolution. Lateral connections between the bottom-up and top-down approaches provide an additional layer of high-resolution features to assist in the localization of objects. Besides that, the loss function is updated by involving the mean squared error loss to aid in the correction of colour and texture distortion, as shown in Equation (2.14),

$$L_G = 0.5L_P + 0.006L_X + 0.01L_{adv} \quad (2.14)$$

where

L_P = MSE loss

L_{adv} = adversarial loss contains both local and global discriminator losses

L_X = same perceptual loss from DeblurGAN network

As a result, the introduction of FPN and MSE loss function in DeblurGAN-v2 greatly improved its training and inference speeds, reducing the required convergence time from 7 to 5 days and enhancing the inference speed by a factor of 100.

In the research conducted by Madam Nimisha et al. (2018), they introduced an unsupervised end-to-end deblurring network that incorporated a GAN model. They claimed that supervised deep-learning approaches relied heavily on massive quantities of paired data, which were both difficult and time-consuming to collect, while unsupervised training strategies could achieve similar performances with unpaired data. From the implementation, their model was able to acquire a robust prior on the domain of clear pictures via adversarial loss. Besides that, they also developed a new convolutional neural network (CNN) module that reblurs the GAN output to match the blurred input,

increasing the GAN's robustness and maintaining picture correlation. They also exploited the blurred image to guide the network and limit the possible outcomes for the clean images to be produced. This can be done through an extra gradient module to impose a gradient error in scale space. Table 2.4 presents an overview of the deblurring models with regard to their hyperparameter settings.

Table 2.4: Comparison of Image Deblurring Methods.

Method	Description	Supervision	Activation Function	Optimization	Regularization	Loss Function	Reference
DeepDeblur, 2017	Deep Multi-scale Network	Supervised	Generator: ReLU, (Tanh for last layer) Discriminator: LeakyReLU	Adam (L-Rate = 5×10^{-5} for the first 300000 epochs, then followed by 5×10^{-6})	Data Augmentation	GAN + MSE	(Nah et al., 2017)
DMPHN, 2019	Stacked Multi-patch Network with Residual Learning and SPM	Supervised	Encoder-Decoder: ReLU	Adam (L-Rate = 10^{-4} for the first 3000 epochs, then followed by $\beta = 0.1$)	-	MSE	(Zhang et al., 2019)
DeepBlurGAN, 2018	Conditional Adversarial Networks with Residual Learning Blocks	Supervised	Generator: ReLU Discriminator: LeakyReLU	Adam (L-Rate = 10^{-4} for the first 150 epochs, then zero for the next 150 epochs)	Dropout ($p = 0.5$)	GAN + Perceptual	(Kupyn et al., 2018)
DeepBlurGAN-v2, 2019	FPN-based Generator and Relativistic Discriminator	Supervised	Generator: ReLU Discriminator: LeakyReLU	Adam (L-Rate = 10^{-4} for the first 150 epochs, then 10^{-7} for the next 150 epochs)	Dropout ($p = 0.5$)	GAN + Perceptual + MSE	(Kupyn et al., 2019)
Unsupervised Deblur, 2018	GAN with Self-Supervision and Gradient Module	Unsupervised	Generator: ReLU (Tanh for last layer) Discriminator: LeakyReLU (Sigmoid for last layer)	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.99$, L-Rate = 5×10^{-4} for the first 100000 epochs, then followed by L-Rate = 10^{-4})	Dropout ($p = 0.2$)	GAN + Reblur + Gradient Note: The weight parameters for the loss function will be updated from $w_{adv} = 1$, $w_{reblur} = 0.01$, $w_{grad} = 0.001$, for the first 100000 epochs, then followed by $w_{adv} = 0.01$, $w_{reblur} = 1$, $w_{grad} = 0.1$.	(Madam Nimisha et al., 2018)

Note: Rectified Linear Unit (ReLU), Learning Rate (L-Rate), Momentum Decay Rate (β)

2.5.2 Image Denoising

In general, many of the techniques and tools developed for denoising can be applied to other types of image inverse problems. This indicates that the majority of the models used for denoising are derived from the other image restoration models. For instance, Zhang et al. (2017) proposed a DnCNN denoising network, which adopted the concept of residual learning to recover a clean image that was obscured by noise. It makes use of a residual network to connect the input image to the output image in a direct manner. Hence, this allows the networks to learn only the residual image without considering the actual content of the images. With this feature, the residual networks bring substantial benefits to the other image restoration techniques as mentioned in several published works (Zhang et al., 2019; Kupyn et al., 2018; Kupyn et al., 2019). Additionally, the batch normalisation strategy was also implemented to improve both training performance and denoising quality.

In order to get a more accurate result, the deep CNN approach appeared to rely on deeper layers rather than shallower ones. Therefore, Tai et al. (2017) proposed recursive and gate units to extract the features adaptively for restoring noisy images. The former is responsible for learning multi-level representations of the current state from various receptive fields; whilst the latter dynamically controls the amount of the present and previous states that should be reserved. Furthermore, Ye et al. (2018) proposed deep convolution framelets that utilized a low-rank Hankel matrix, which convolves local and nonlocal bases to illustrate the correlation between deep learning and signal processing. However, the CNNs proposed earlier are associated with higher processing costs and memory usage, thus making them impractical in reality. Hence, as an example, the application of dilated convolutions in computed tomography (CT) image denoising had been proposed by Gholizadeh-Ansari et al. (2020). Instead of using conventional convolution, they developed a deep neural network that leveraged dilated convolutions with various dilation rates to capture more contextual information with low computational complexity. Additionally, an integrated simple edge detection layer comprised of Sobel operators enhances the computation of a 2D gradient of image intensity. As a result, the receptive field can be expanded without incurring additional costs, and the depth of the network can be shrunk without sacrificing performance.

In the real world, all the noise signals do not behave the same and images can be randomly distorted. This resulted in the development of multiple models with different algorithms to deal with various types and levels of noise disturbances. This has led to the consideration of blind denoising methods. As a result, Zhang et al. (2018) proposed Fast and Flexible Denoising CNN (FFDNet), which adopted a configurable noise level input. FFDNet operated on downsampled sub-images while maintaining a decent trade-off between denoising performance and inference speed. By selecting a non-uniform noise level map, their model is able to handle a wide variety of noise levels with only a single CNN, and filter out the noise with spatial variations. Other than that, soft shrinkage was adopted by Isogawa et al. (2018) in their image denoising mechanism to modify the noise level adaptively, resulting in the proposal of Soft Shrinkage CNN (SCNN). In other words, their CNN model can be instantly tuned to match various noise levels of any given input image. Table 2.5 shows PSNR performance results on BSD68 datasets for different denoising models. Table 2.6 presents an overview of the denoising models as well as their hyperparameter settings.

Table 2.5: Result of PSNR Performance between FFDNet with Different Models on BSD68 Datasets (Source: Zhang et al., 2018; Isogawa et al., 2018).

Noise Level	BM3D	DnCNN	FFDNet	SCNN
15	31.07	31.72	31.63	31.48
25	28.57	29.23	29.19	29.03
50	25.62	26.23	26.29	26.08

Note: Block-Matching and Three-Dimensional (3D) Filtering (BM3D), is a classical method that groups similar image blocks into 3D arrays and applies collaborative filtering techniques to reduce noise.

Table 2.6: Comparison of Image Denoising Methods.

Method	Description	Supervision	Activation Function	Optimization	Regularization	Loss Function	Reference
DnCNN, 2017	Deep CNN with Residual Learning	Supervised	ReLU (for first Conv layer) ReLU + BN (for middle Conv layers) None (for last Conv layers)	SGD (L-Rate = 10^{-1} for the first 50 epochs, then followed by 10^{-4})	-	MSE	(Zhang et al., 2017)
MemNeT, 2017	Deep CNN with Recursive and Gate Units	Supervised	ReLU + BN	SGD (L-Rate = 10^{-1} and then reduced by a factor of 10 for every 20 epochs)	Data Augmentation + Modified L2-norm	MSE	(Tai et al., 2017)
Multi-resolution DCF, 2018	Deep Convolutional Framelets with Hankel Matrix	Supervised	ReLU + BN (only for first Conv layer)	Adam ($\beta_1 = 0.9$. L-Rate = 10^{-4} , then reduced half for every 50 epochs, until L-Rate = 10^{-5})	Data Augmentation	MSE	(Ye et al., 2018)
DRL Denoising, 2020	CNN with Dilated Convolutions and Edge Detection Layer	Supervised	ReLU (for first Conv layer) ReLU + BN (for middle Conv layers) None (for last two Conv layers)	Adam (L-Rate = 10^{-3} for the first 20 epochs, then 10^{-4} for the next 20 epochs)	-	MSE + Perceptual	(Gholizadeh-Ansari et al., 2020)
FFDNet, 2018	Blind Denoising CNN with Varying Noise Level	Unsupervised	ReLU (for first Conv layer) ReLU + BN (for middle Conv layers) None (for last Conv layer)	Adam (L-Rate = 10^{-3} and then reduced to 10^{-4} when the training error stop decreasing. Additional 50 epochs with L-Rate = 10^{-6} to fine tune the model)	Data Augmentation	MSE	(Zhang et al., 2018)
SCNN, 2018	Blind Denoising CNN with Soft Shrinkage	Unsupervised	Soft Shrinkage (for feature extraction layers) Soft Shrinkage + BN (for feature conversion layers)	Adam (L-Rate = 10^{-3})	-	MSE	(Isogawa et al., 2018)

Note: Stochastic Gradient Descent (SGD), Batch Normalization (BN)

2.5.3 Super-Resolution

Super-resolution can be categorized as either single-image or multi-image, depending on the number of input images. Single-image super-resolution, also known as SISR, will be discussed in this section since it is widely known for its high efficiency. However, SISR is a typical ill-posed issue. This is because a single low-resolution input might be associated with a large number of high-resolution images, as well as the mapping solution between these two elements is typically intractable.

A simple model named Super-Resolution Convolutional Neural Network (SRCNN) was proposed by Dong et al. (2016) to perform the super-resolution task. The network consists of only three layers that served for patch extraction and representation, non-linear mapping, and reconstruction, respectively. The authors performed some ablation studies regarding the architecture of SRCNN. First of all, the low-resolution input of SRCNN was obtained through bicubic interpolation, which leads to inaccurate estimates and additional processing time. To overcome this problem, they suggested to replace this algorithm of low-resolution input. Besides that, they also claimed that the super-resolution performance will vary depending on the width and depth of the CNN architecture used.

In 2016, Shi et al. proposed Efficient Sub-Pixel Convolutional Neural Network (ESPCN), which utilized a sub-pixel convolution layer. They argued that non-linear convolution feature extraction should be used in low-resolution space rather than high-resolution space. Hence, all of the low-resolution features are fed into the network that will be utilized at the last layer for generating the high-resolution output. By doing so, they proposed expanding the channel of upscaling filters to store additional pixels, instead of directly enlarging input feature maps to enhance the resolution as the bicubic filter does. Therefore, a smaller size of the input kernel is sufficient enough since the upscaling process occurs in the channel dimension. This can reduce computational and memory complexity while preserving a certain contextual region.

Deep CNN networks have been shown to achieve superior performance in many image restoration tasks. Hence, Kim et al. (2016a) came up with Very Deep Super-Resolution (VDSR), which was the first deep CNN model to perform SISR tasks. Similarly, the VDSR utilizes the same bicubic

low-resolution input, but the novelty is different scale factors of low-resolution bicubic are adopted during the training phase. In addition, they intervened the residual learning block between bicubic interpolation and high-resolution output to accelerate the convergence of the model training. However, the proposed complicated deep model network requires additional gradient clipping to overcome vanishing gradient problems and a high initial learning rate to accelerate convergence. Therefore, the same author further proposed Deep-Recursive Convolutional Network (DRCN), which adopted recursive learning and skip-connection to ease the difficulty of training (Kim et al., 2016b). They argued that adding another weight layers resulted in more variables, which made the model too large to be retrieved and stored, and increased the risk of overfitting. Instead, they reconstructed a high-resolution image using the feature maps generated at the end of each recursion level, and integrated all the predictions to produce a more robust final prediction. Skip-connections are integrated to prevent gradient vanishing, which is superior to the gradient clipping technique used in VDSR. In terms of performance, both the VDSR and DRCN models produce results that are very close to one another.

In 2018, the Zero-Shot Super Resolution (ZSSR) model, which adopted image-specific learning in CNN, was proposed by Shocher et al. As the name suggested, zero-shot is a method that does not utilise any prior image examples or pre-training. Hence, the model only requires the test images by extracting the relevant internal patches for the training. Contrary to external-example SISR approaches, a small CNN is enough to build the ZSSR model due to the accessibility of downscaling of test images in image-specific learning. After observing the success of VDSR, which used a small-scale model for training, the researchers of the ZSSR model came up with the notion of using a similar approach to gather more small scaled internal training pairs for training large-scale models. Besides that, they also adopted the concept of geometric self-ensemble, which produced eight different outputs with various rotation angles and flipping directions. The resulting images were then combined to generate the median image through the back-projection algorithms. However, ZSSR has a downside as it increases runtime during the testing phase. Table 2.7 depicts an overview of the super-resolution models including their hyperparameter settings.

Table 2.7: Comparison of Super-resolution Methods.

Method	Description	Supervision	Activation Function	Optimization	Regularization	Loss Function	Reference
SRCNN, 2016	Simple CNN with for Super-Resolution	Supervised	ReLU	SGD with standard backpropagation (L-Rate = 10^{-1} for the first 2 layers, and 10^{-3} for the last layer)	-	MSE	(Dong et al., 2016)
ESPCN, 2016	Sub-Pixel CNN with Upscaling Filters	Supervised	Tanh	SGD (L-Rate = 10^{-2} and then reduced to 10^{-4} when the loss function value smaller than a threshold)	-	MSE	(Shi et al., 2016)
VDSR, 2016	Very Deep CNN with Adjustable Gradient Clipping and Residual Learning	Supervised	ReLU	SGD (L-Rate = 10^{-1} , then reduced by a factor of 10 for every 20 epochs, until 80 epochs)	Data Augmentation + L2-norm	MSE	(Kim et al., 2016a)
DRCN, 2016	Deep CNN with Skip Connection and Recursive Supervision	Supervised	ReLU	SGD (L-Rate = 10^{-2} , then reduced by a factor if 10 if validation error remains constant for 5 epochs, until L-Rate = 10^{-6})	L2-norm	MSE	(Kim et al., 2016b)
ZSSR, 2018	Zero-Shot CNN with Small Image-Specific and Geometric Self-ensemble Techniques	Unsupervised	ReLU	Adam (L-Rate = 10^{-2} , then reduced by a factor if 10 if the reconstruction error greater than the slope of linear fit by a factor, until L-Rate = 10^{-6})	L1-norm	MAE	(Shocher et al., 2018)

Note: Mean Absolute Error (MAE)

2.5.4 Investigation of Deep Learning Approaches to Image Restoration

2.5.4.1 Image Deblurring

Figure 2.25 displays the deblurring results on CSet9 datasets using DL model with all the results recorded in Table 2.8 accordingly. It can be observed that the recovered images have outstanding performance in removing blurs.

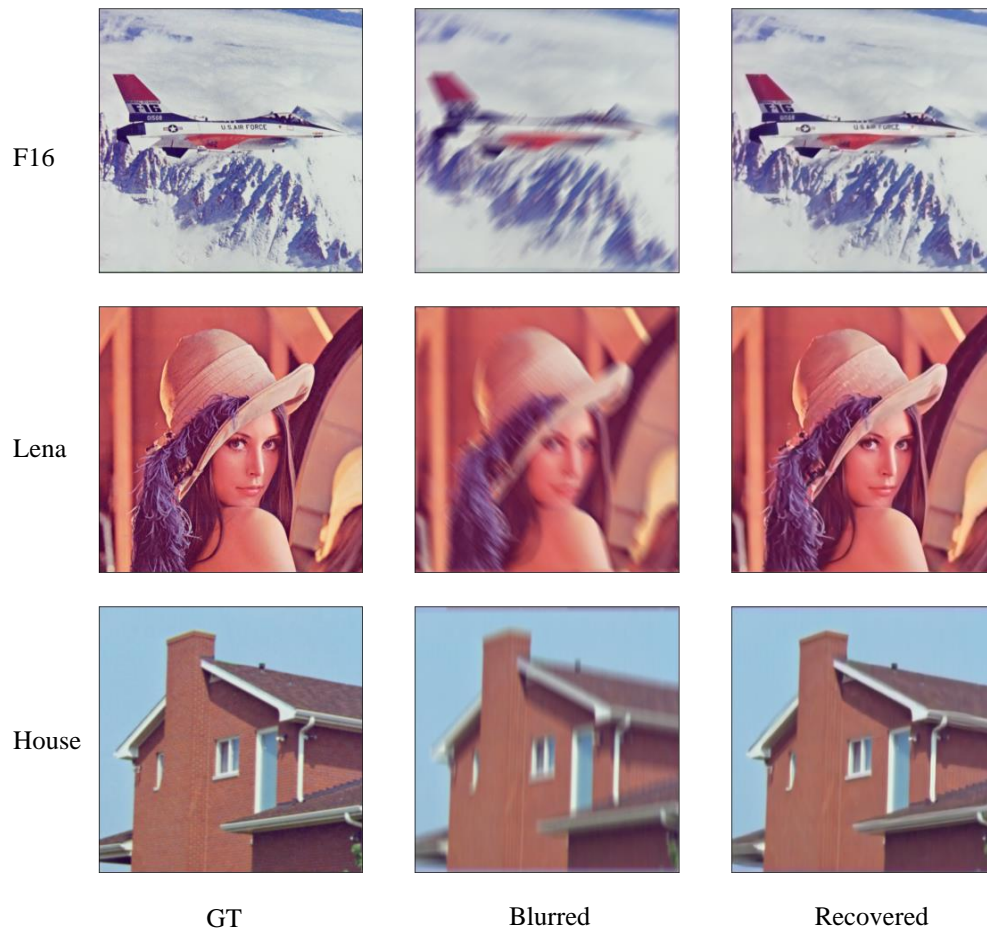


Figure 2.25: Sample of Deblurring Results of DL Approaches on CSet9 Datasets.

Table 2.8: Performance Comparison on CSet9 Datasets for Deblurring Results Using DL Approaches.

	Blurred Image			Recovered Image		
	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑
F16	0.0831	20.8300	0.7900	0.0448	26.1899	0.9280
Lena	0.0722	22.7267	0.7761	0.0359	28.8007	0.9025
House	0.0589	24.1314	0.8863	0.0349	28.6675	0.9349

Note: The symbols \downarrow and \uparrow indicate that lower and higher values, respectively, represent better performance for the corresponding metric.

2.5.4.2 Image Denoising

Figure 2.26 shows the denoising results on CSet9 datasets with a DL model, and all the results are tabulated in Table 2.9. It is worth noting that the denoising algorithm produces images of exceptional quality, with a significant reduction in noise levels.

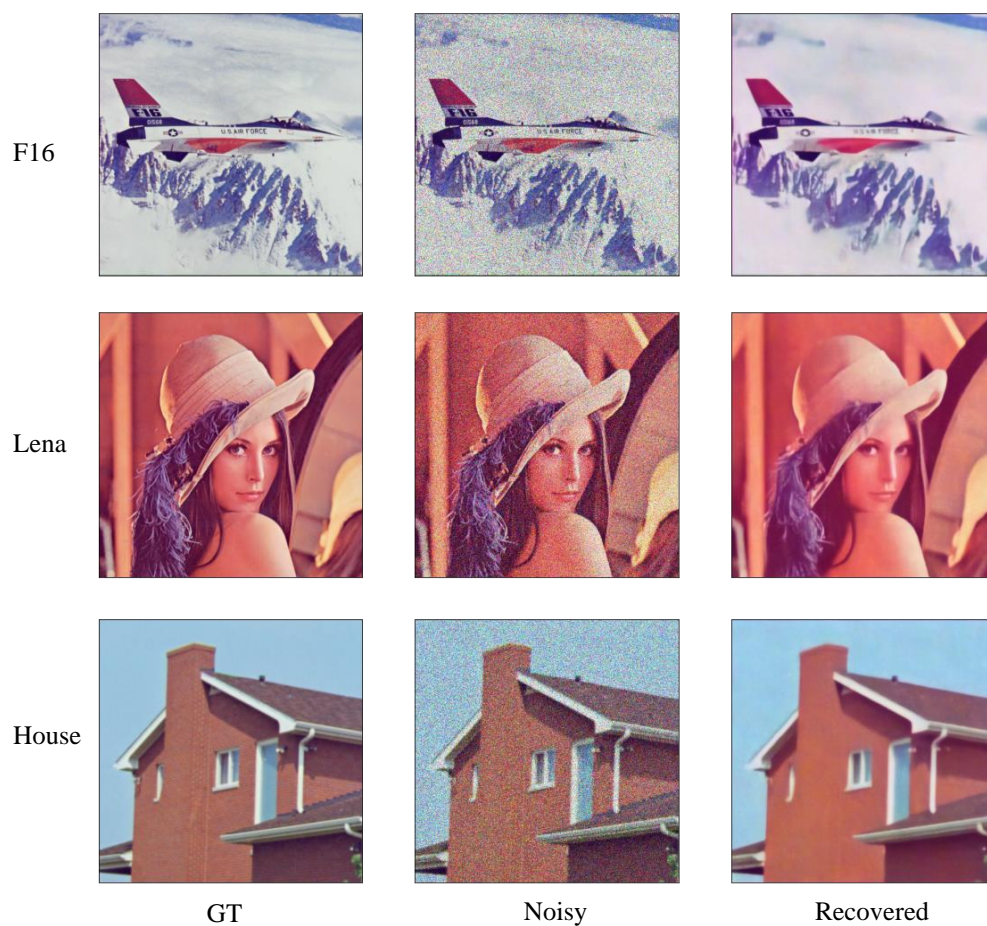


Figure 2.26: Sample of Denoising Results of DL Approaches on CSet9 Datasets.

Table 2.9: Performance Comparison on CSet9 Datasets for Denoising Results Using DL Approaches.

	Noisy Image			Recovered Image		
	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑
F16	0.2133	12.6347	0.1874	0.0655	22.8866	0.8946
Lena	0.1685	15.3647	0.2204	0.0568	24.8124	0.8715
House	0.1928	13.8427	0.1477	0.0484	25.8516	0.9391

Note: The symbols ↓ and ↑ indicate that lower and higher values, respectively, represent better performance for the corresponding metric.

2.5.4.3 Super-Resolution

Figure 2.27 shows the super-resolution results on CSet9 datasets using DL model, and all the results are tabulated in Table 2.10. The super-resolution results are impressive, clearly demonstrating the effectiveness of the DL model in generating high-quality images with finer details.

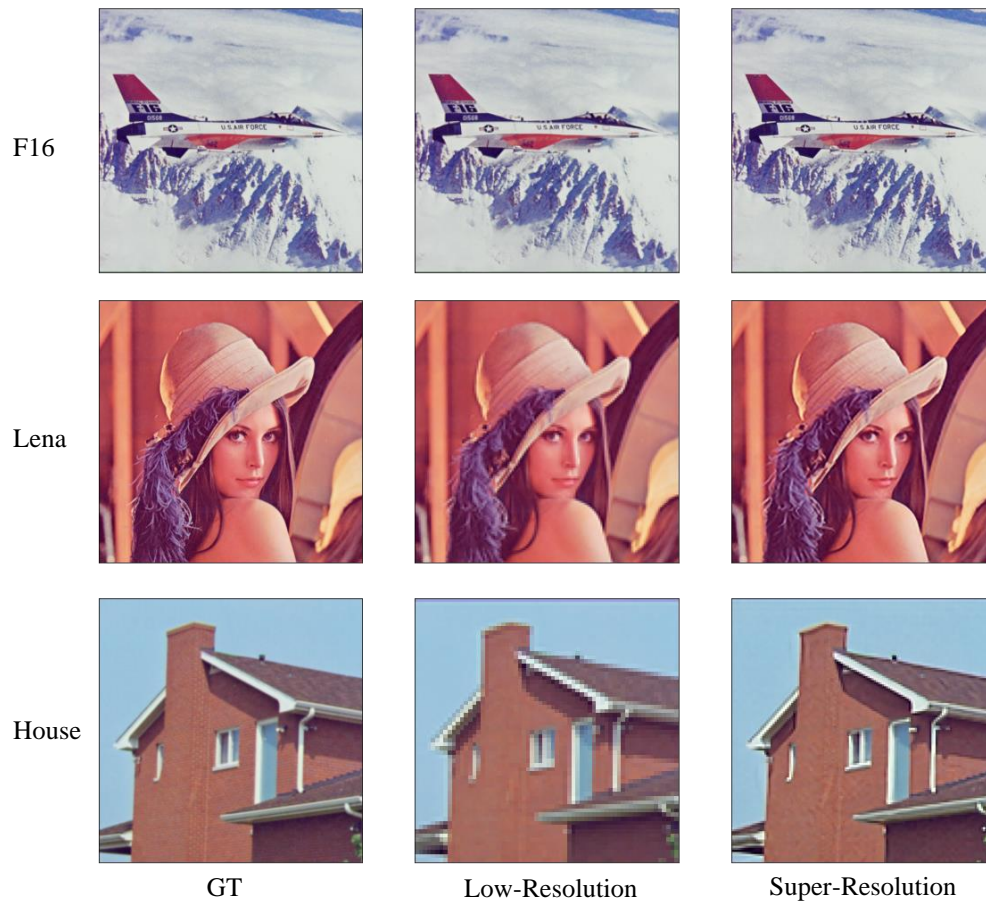


Figure 2.27: Sample of Super-Resolution Results of DL Approaches on CSet9 Datasets.

Table 2.10: Performance Comparison on CSet9 Datasets for Super-Resolution Results Using DL Approaches.

	Super-Resolution Image		
	RMSE ↓	PSNR ↑	SSIM ↑
F16	0.0205	32.9751	0.9780
Lena	0.0317	29.8713	0.9187
House	0.0512	25.3688	0.8912

Note: The symbols ↓ and ↑ indicate that lower and higher values, respectively, represent better performance for the corresponding metric. Evaluation metrics cannot be performed on low-resolution images due to the unmatched pixel size between ground truth and low-resolution images.

2.5.5 Summary

From the research papers that had been discussed above, it can be deduced that efficiency can be improved by using a learning-based approach. Many benchmark datasets show that deep learning-based methods greatly outperform conventional alternatives. Besides that, high efficiency is achieved when deep learning algorithms are implemented on parallel processing units such as graphic processing units (GPUs) rather than central processing units (CPUs). Nevertheless, deep learning-based algorithms contribute to high computational costs, thus making them challenging to implement in real-time processing. As an example, matrix processing requires more computer hardware, such as GPUs and random-access memory (RAM) which are costly and difficult to be obtained. Last but not least, enormous training datasets are necessary for deep learning CNNs, yet they are difficult to be collected, labelled, and may not even be a good fit for real-world scenarios.

2.6 Deep Image Prior (DIP) Approaches to Image Restoration

2.6.1 Overview

In the DIP model, a neural network is randomly initialised and then processed using only the distorted image. After that, the network will try to generate an image that is similar to the distorted one. Noise naturally exhibits high impedance so the model prefers to learn the original pixel over the noise. Hence, the early stopping technique is applied to prevent the model from overfitting. It

is used to terminate the reconstruction process to get rid of the degradation details including jagged edges, artifacts and others. Due to the lack of training datasets, the complexity of the DIP network architecture plays an important role in determining the performance of the image restoration process (Ulyanov et al., 2018). Figure 2.28 illustrates the overview structure of the DIP model.

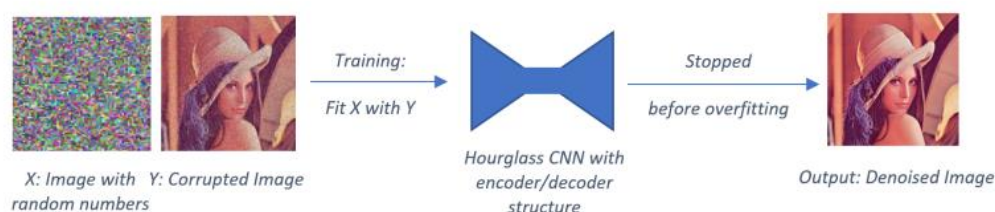


Figure 2.28: Overview of Deep Image Prior Model Network (Source: Ulyanov et al., 2018).

The DIP model can be applied to several image restoration tasks such as denoising, deblurring, inpainting, flash no-flash reconstruction and others. This can be accomplished by controlling over aspects of the network's structure, such as the number of hidden layers, and hyperparameters such as stopping criteria and learning rate. The sample results will be provided in the following subsections.

2.6.2 Investigation of Deep Image Prior to Image Restoration

2.6.2.1 Image Denoising

For the image denoising results in Figure 2.29, it can be observed that the DIP model exhibits remarkable noise reduction performance, as indicated by the evaluation metrics in Table 2.11.

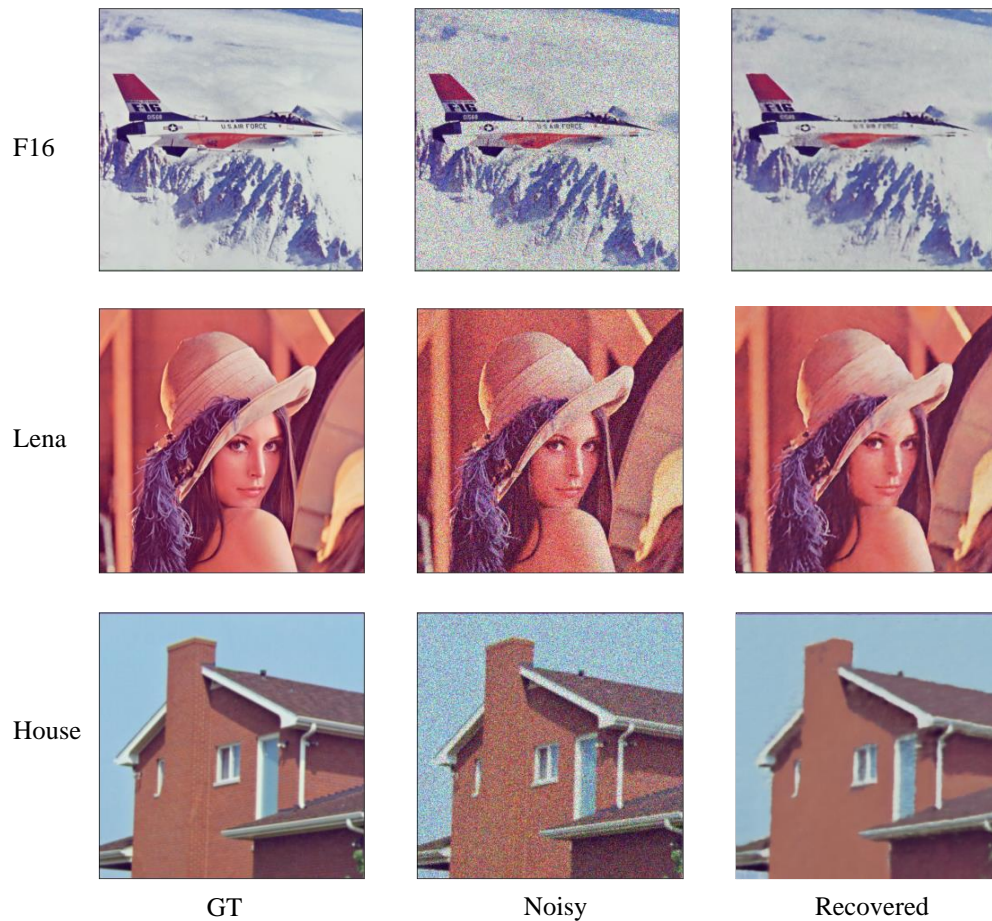


Figure 2.29: Sample of Denoising Results of DIP Model on CSet9 Datasets.

Table 2.11: Performance Comparison on CSet9 Datasets for Denoising Results Using DIP Model.

	Noisy Image			Recovered Image		
	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑
F16	0.2133	12.6347	0.1874	0.0599	23.6691	0.8798
Lena	0.1685	15.3647	0.2204	0.0421	27.5219	0.8707
House	0.1928	13.8427	0.1477	0.0353	29.0368	0.9497

Note: The symbols ↓ and ↑ indicate that lower and higher values, respectively, represent better performance for the corresponding metric.

2.6.2.2 Super-Resolution

The super-resolution results displayed in Figure 2.30 indicate that the DIP model can effectively enhance the resolution of low-quality images, achieving impressive performance metrics as listed in Table 2.12.

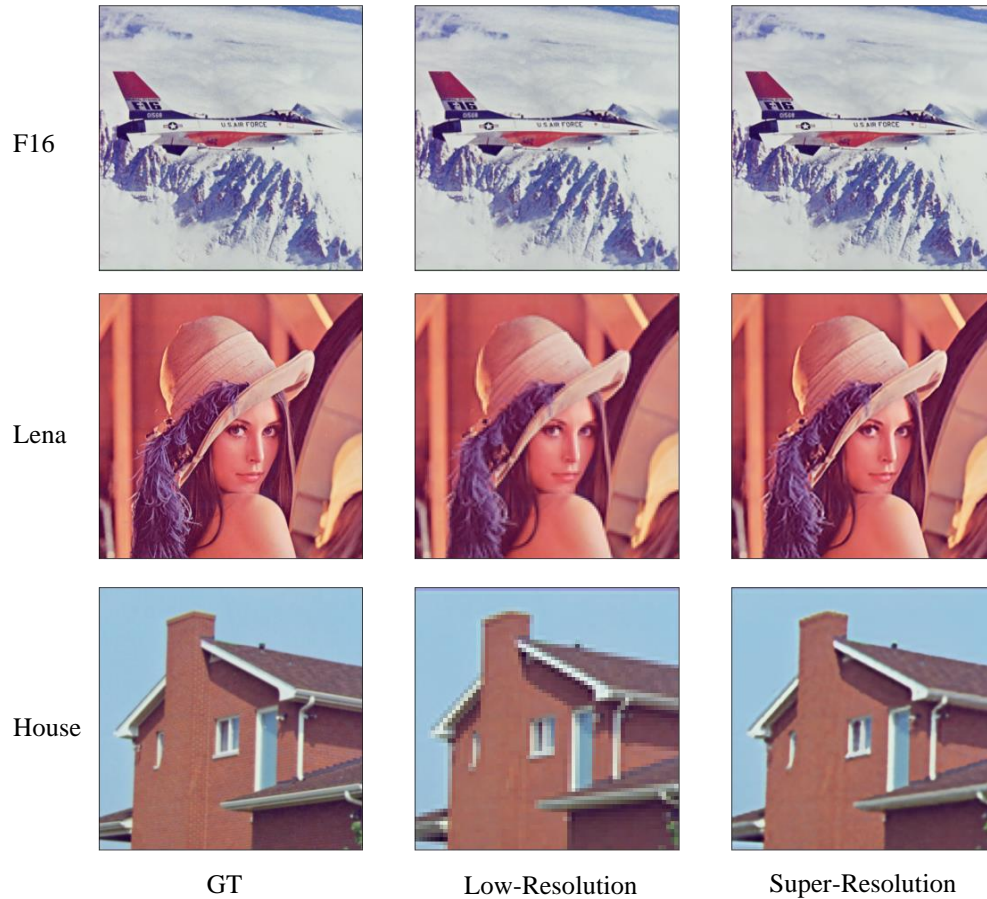


Figure 2.30: Sample of Super-Resolution Results of DIP Model on CSet9 Datasets.

Table 2.12: Performance Comparison on CSet9 Datasets for Super-Resolution Results using DIP Model.

	Super-Resolution Image		
	RMSE ↓	PSNR ↑	SSIM ↑
F16	0.0243	31.4889	0.9666
Lena	0.0317	29.8725	0.9097
House	0.0294	30.1842	0.9334

Note: The symbols ↓ and ↑ indicate that lower and higher values, respectively, represent better performance for the corresponding metric. Evaluation metrics

cannot be performed on low-resolution images due to the unmatched pixel size between ground truth and low-resolution images.

2.6.2.3 Impainting

Inpainting process refers to the technique of reconstructing missing or damaged parts of an image, video, or any other signal. It involves filling in the missing or damaged regions using the information present in the surrounding areas or using some other prior information. Figure 2.31 displays the inpainting result on CSet9 datasets in which all the results are recorded in Table 2.13. It can be observed that the DIP model effectively restores the missing parts of the images, resulting in visually pleasing results with high PSNR and SSIM scores.



Figure 2.31: Sample of Impainting Results of DIP Model on CSet9 Datasets.

Table 2.13: Performance Comparison on CSet9 Datasets for Impainting Results using DIP Model.

	Corrupted Image			Recovered Image		
	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑
F16	0.2470	11.3617	0.8717	0.0316	29.2184	0.9599
Lena	0.1299	17.6263	0.8383	0.0188	34.3975	0.9535
House	0.1132	18.4683	0.9088	0.0098	39.7476	0.9858

Note: The symbols ↓ and ↑ indicate that lower and higher values, respectively, represent better performance for the corresponding metric.

2.6.3 Summary

However, there are several problems with the DIP model. Since the network needs to be run specifically to replicate each image while simultaneously updates the network's hyperparameters, thus the time required to produce a single image will be extremely long. On top of that, distinct hyperparameter settings are required for every image restoration task, and it has to be fine-tuned manually, which is impractical in reality.

2.7 Overall Summary

There are many approaches to recover an image in terms of quality, which are conventional, DL, and DIP methods. From the literature review, each approach has its own pros and cons. Some of the models only work for certain image restoration tasks, which results in a variety of models with different algorithms. This can be overcome by employing DL methods with the presence of datasets. However, the training process took a long time due to a large number of datasets is required. Thus, the DIP model can be useful for image restoration tasks where dataset is not available. With the proper selection of hyperparameters and an appropriate early stopping criterion, satisfactory results can be achieved without the requirement of pre-learning or extensive datasets. Nevertheless, both methods share the same issue, whereby significant resources and time are needed to modify the network's architecture and incorporate additional techniques in order to achieve the desired performance.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The main objective of this study is to propose an innovative approach for detecting the optimal stopping point in the image restoration process using the DIP model. To effectively carry out this study, the necessary hardware devices and software tools such as GPUs, deep learning frameworks, and Python libraries were utilized. The effectiveness of the proposed algorithm was evaluated by comparing its performance with the classical and iterative image restoration models, as well as recent deep learning models.

3.2 Work Plan

Figure 3.1 shows the general process flow of the project, outlining the major stages involved in achieving the objectives. The first stage is an analysis of image restoration in terms of image models and degradation models, as depicted in Figure 2.1. This is followed by a comprehensive literature review on image restoration tasks from a variety of perspectives, as shown in Figure 2.2. The review includes an in-depth study of journals, publications, conference proceedings, electronic books, and other sources of information. After reviewing the related literature, the limitations of the image restoration tasks are identified. Hence, the aim and objectives are defined based on the listed problem statements.

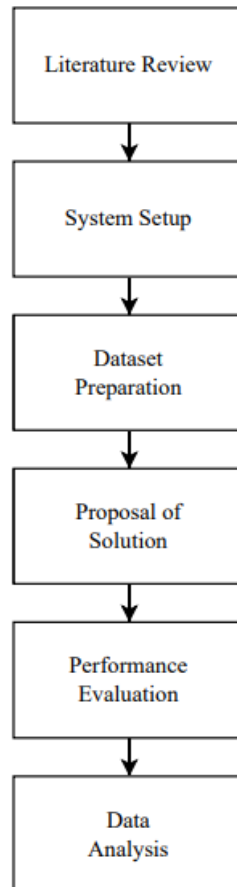


Figure 3.1: General Process Flow.

Subsequently, system setup focuses on acquiring and preparing the necessary hardware devices and software tools required to effectively carry out the project. Once the system is set up, the next stage involves preparing the test images by selecting appropriate test images and applying different types of degradation to them, such as low-resolution or noise, to simulate real-world scenarios. Accordingly, solution is proposed to detect the optimal stopping point in the image restoration process using the DIP model.

Following the proposal of the solution, the performance evaluation is conducted, which involves validation on test images and performance analysis in terms of image quality metrics, such as PSNR and SSIM. In addition, the proposed algorithm is compared with classical and iterative image restoration models, and recent deep learning models, to evaluate its effectiveness. Finally, all data are analyzed and discussed. Conclusions and recommendations for future research are made.

Figure 3.2 shows the FYP1 gantt chart. The project details were discussed with the supervisor at the beginning of the first semester. Next, a general project overview regarding the background of image restoration was conducted. Subsequently, a project introduction and report overview were done for the subsequent two weeks. Then, an in-depth study was performed to analyse different image restoration tasks. The project flow and the methodology were planned to investigate image restoration in terms of implementation methods and performance metrics. Preliminary results tabulation, report writing, and presentation preparation were carried out as planned in the following weeks.

No.	Project Activities	Planned Completion Date	Weeks														
			W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	
1.	Discuss the project details.	2022-06-17	█														
2.	Formulate general project overview.	2022-06-24		█													
3.	Write the project introduction and the report overview.	2022-07-08			█	█											
4.	Find literature, review and compile relevant studies into the report.	2022-08-19			█	█	█	█	█	█	█	█					
5.	Plan the project flow and write the methodology.	2022-08-19								█	█	█					
6.	Record the preliminary results, report writing, and presentation preparation.	2022-09-02										█	█	█	█	█	

Figure 3.2: FYP1 Gantt Chart.

Figure 3.3 shows the FYP2 gantt chart. The project was further discussed with the supervisor to schedule the time frame for each project activity. To achieve the project objective, a novel algorithm was proposed based on the DIP model, with a focus on improving the image restoration process. A tabulation of the data was performed for different image restoration tasks, and the results were then compared to other proposed solutions. Subsequently, the code was optimized by removing unnecessary building processes. Towards the end of the trimester, poster was prepared followed by report writing and presentation preparation.

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
1.	Project Planning	2023-02-10	■	■												
2.	Programming and Debugging	2023-03-10		■	■	■	■	■								
3.	Project Analysis and Data Tabulation	2023-03-24		■	■	■	■	■	■	■						
4.	Code Optimization	2023-03-24							■	■						
5.	Poster Planning and Preparation	2023-04-14							■	■	■	■	■			
6.	Report Writing and Presentation Preparation	2023-04-21								■	■	■	■	■	■	■

Figure 3.3: FYP2 Gantt Chart.

3.3 Implementation

The image restoration tasks in this project were performed using the Python environment, which provides access to a wide range of useful library modules such as OpenCV, scikit-image, and PyTorch. These tools were utilized to process the images for the preparation of image restoration and evaluate the performance of different restoration techniques.

In this project, Python codes were executed on both laptop and Google Colab. The proposed algorithm was developed on a laptop with an anaconda environment to ensure the installed Python modules would not interfere with the global Python settings. Locally installed frameworks such as TensorFlow, PyTorch, and Compute Unified Device Architecture (CUDA) were required to make the laptop's GPU processing accessible for deep learning implementation. On the other hand, Google Colab provides free access GPU in the browser and utilizes the Jupyter notebook for Python code execution. Therefore, cloud computing services from Google Colab were utilized to execute other restoration methods from Github sources for comparison purposes. One advantage of using Google Colab is that modules and frameworks can be installed directly without the need to create a virtual environment as the system resets once the allocated runtime duration is over. Table 3.1 shows the specification of local platform and Google Colab for running the Python codes.

Table 3.1: Specifications of Local Platform and Google Colab in the Project.

	Local Platform	Google Colab
CPU	Intel Core i5-8300H CPU @ 2.30 GHz	Intel Xeon @ 2.20 GHz
CPU Cores	4	2
Memory	16 GB	12 GB
GPU	NVIDIA GeForce GTX1060 6GB	NVIDIA Tesla K80 12GB
Storage	512 GB SSD	100 GB SSD
OS	Windows 10	Ubuntu 18.04 LTS

Note: Operating System (OS)

3.4 Performance Metrics

The performance metrics used in the analysis study of the image restoration process are RMSE, PSNR, and SSIM as shown in Table 3.2.

Table 3.2: Evaluation Metrics of Image Quality.

Performance Metrics	Description	Equation
Root Mean Square Error (RMSE)	RMSE measures the square root of the cumulative squared error between the original and the degraded image. Lower values indicate higher image quality after restoration.	RMSE is calculated as follows, given a $m \times n$ monochromatic noise-free image I and the noisy estimation of K : $RMSE = \sqrt{\frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2} \quad (3.1)$
Peak signal-to-noise ratio (PSNR)	PSNR is defined as the strength of an image relative to the power of the noise that degrades its representation. In general, a greater score indicates a higher quality reconstructed image.	PSNR is further derived from RMSE, the equation is derived as, $PSNR = 10 \log_{10} \left(\frac{MAX_I}{RMSE} \right)^2 \quad (3.2)$

where
 MAX_I = the highest possible signal intensity in the actual image with a formula of $2^{B_s} - 1$, given that B_s is bits per sample.

Structural Similarity Index Measure (SSIM)	SSIM measures how much amount of quality is lost in an image due to factors such as data compression or transmission losses. The value ranges from 0 to 1, 1 means there is a perfect match between the original and reconstructed images.	SSIM is calculated based on two windows named x and y that share a common size of $N \times N$ is determined by,
---	--	--

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (3.3)$$

where

μ_x = average of window x

μ_y = average of window y

σ_x^2 = variance of window x

σ_y^2 = variance of window y

σ_{xy} = covariance of windows x and y

$c_1 = (k_1 R_{dynamic})^2$

$c_2 = (k_2 R_{dynamic})^2$

given that the $R_{dynamic}$ is the pixel-value dynamic range with a formula of $2^{\text{number of bits per pixel}} - 1$. By default, the values for k_1 and k_2 are 0.01 and 0.03, respectively.

3.5 Test Image Preparation

The CSet9 dataset was pre-processed to obtain test images for this study. The test set can be divided into two parts: denoising and super-resolution. The denoising test set comprises clean images and their corresponding noisy images, each generated with different types of noise and intensity levels. The super-resolution test set contains high-resolution (HR) images and their respective low-resolution (LR) images with different scaling factors. The purpose of utilizing these test images is to assess the effectiveness of the proposed early stopping algorithm for the DIP model in accomplishing denoising and super-resolution tasks. The following subsections provide details on the noise type, intensity level, and scaling factor used for each test image.

3.5.1 Denoising

Three common types of noise, namely Gaussian, Speckle, and Shot noise, with low and high levels for each type, were generated in this project following the rules and approaches outlined by previous works (Hendrycks and Dietterich, 2019; Wang et al., 2021). Impulse noise was not included in the project due to the limitations of the original DIP neural network. Furthermore, modifying the DIP neural network was not the main focus of this project. The noise can be modelled by using Numpy's random functions. The following is a brief explanation of the noise types and their corresponding parameters.

- (i) Gaussian noise: Additive noise with a mean of 0 and a variance of either 0.12 or 0.18 for low or high levels, respectively. The noise is generated using NumPy's `random.normal` function.
- (ii) Speckle noise: Multiplicative noise with a mean of 0 and a variance of either 0.20 or 0.35 for low or high levels, respectively. The noisy pixel is $x(1 + \varepsilon)$, where ε is 0-mean Gaussian with a variance level, and the noise is generated using NumPy's `random.normal` function.
- (iii) Shot noise: Poisson noise with a rate λx , where λ is 25 or 12 for low or high levels, respectively. The noisy pixel is Poisson distributed, and the noise is generated using NumPy's `random.poisson` function.

Table 3.3 summarizes the specific settings and corresponding parameter values used for generating the noise, along with the code snippet for each noise type. Figure 3.4 presents a sample of clean images alongside noisy images affected by Gaussian, Speckle, and Shot noise at both low and high noise levels.

Table 3.3: Parameters for Different Types of Noise.

Noise Type	Parameters	Code Snippet
Gaussian	Mean = 0 Variance = 0.12 (low) & 0.18 (high)	1. <code>noise = np.random.normal(mean, variance, img.shape)</code> 2. <code>noisy_img = img + noise</code>
Speckle	Mean = 0 Variance = 0.20 (low) & 0.35 (high)	1. <code>noise = np.random.normal(mean, variance, img.shape)</code> 2. <code>noisy_img = img + (img * noise)</code>
Shot	Rate = 25 (low) & (12) high	1. <code>noisy_img = np.random.poisson(rate * img, img.shape)</code> 2. <code>noisy_img = noisy_img / float(rate)</code>

Note: "np" refer to the NumPy library module, "img" refer to the input image, and "shape" refer to the shape of the image (e.g., 512×512).

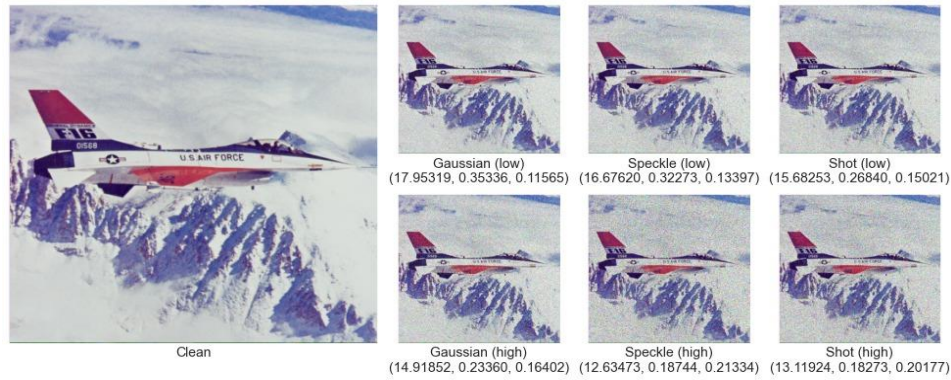


Figure 3.4: Examples of Clean and Noisy Images with Different Types of Noise and Intensity Level: Gaussian, Speckle, and Shot Noises. Note: The performance metrics was measured in the form of (PSNR, SSIM, RMSE).

3.5.2 Super-resolution

To prepare test images for super-resolution, the original high-resolution image is all that is needed. The downsampled low-resolution image can be obtained through the function provided from the original DIP work (Ulyanov et al., 2018). The function loads the image and optionally resizes it to a new size using the specified scaling factor (SF). For instance, an image with a size of 512×512 can be resized to 128×128 with a SF of 4, which is obtained by dividing the original dimensions by the SF. The resulting LR image is returned as NumPy arrays, which will be used for super-resolution later. Figure 3.5 presents a sample of downsampled images with their respective SF of 2, 3, 4 and 8.



Figure 3.5: Examples of Downsampled Images with Different Scaling Factors for Super-Resolution. Note: SF refer to Scaling Factor.

3.6 DIP Model

DIP model is recently developed for image restoration tasks (Ulyanov et al., 2018). The DIP model utilizes the structure of CNNs to restore images without requiring any prior knowledge of the image formation model or degradation process. Instead, the network is trained using only the degraded image itself, which allows it to learn the underlying distribution of the original image from the degraded data.

To implement the DIP model for image restoration, a CNN with an encoder-decoder architecture is constructed. The encoder is responsible for reducing the input image to a lower-dimensional representation, while the decoder maps the representation back to the original image domain. The CNN network parameters are randomly initialized and optimized using Adam optimizer and MSE loss function. The optimization is done with respect to the network parameters and the restored image.

From a higher-level perspective, a set of features will be generated at each encoder layer, which are then concatenated with the corresponding decoder features. After concatenating the features, they are fed into a series of convolutional layers, resulting in the final output image. The inclusion of skip connections in the DIP model plays a vital role in preventing information loss during the encoding process. It allows information to be transmitted directly from the encoder to the decoder, ensuring that important image features are preserved in the restored image.

The general process of the DIP model to restore an image can be formulated as follows:

$$x^* = \arg \min_x |\mathcal{A}(x) - y|_2^2 + \lambda R(x) \quad (3.4)$$

where

x = restored image

y = degraded image

\mathcal{A} = degradation operator

λ = regularization parameter

$R(\cdot)$ = a regularizer that encourages the restored image to be smooth or have other desirable properties.

Equation (3.4) represents the optimization problem aimed at restoring an image x from a degraded image y . The degradation operator $\mathcal{A}(\cdot)$ models the degradation process that has affected the image and can be customized to suit different restoration tasks. The objective is to find the restored image x^* that minimizes the difference between the degraded image y and the reconstructed image obtained from the degradation operator $\mathcal{A}(x)$. A regularization term $R(\cdot)$ is also incorporated into the objective function to encourage the restored image to be smooth or possess other desired properties. The regularization parameter λ balances the trade-off between fidelity to the degraded image and the regularization term. By solving this optimization problem, the equation can produce a restored image that closely resembles the original image as compared to the degraded image.

Overall, the DIP model, including the skip connections, provides an effective way to restore images by leveraging the power of deep learning techniques. The DIP model employs an encoder-decoder architecture with skip connections and optimizes the loss function via gradient descent to produce high-quality restored images from degraded inputs. Nevertheless, the DIP is subjected to overfitting issue due to the optimization process based on the reconstruction loss itself. As the iteration continues, the model may start to incorporate the noise and artifacts present in the degraded input into the restored image, resulting in overfitting. To prevent this, early stopping is often used to halt the optimization process before overfitting occurs. However, there was no method or algorithm to determine the appropriate stopping point for the DIP from the original author (Ulyanov et al., 2018).

3.7 Early Stopping Detection Method

The study by Wang et al. (2021) discussed the implementation of early stopping in the DIP model. This was achieved by monitoring the trend of the running variance of the reconstruction sequence, leading to the proposal of early stopping using exponential moving variance (ES-EMV). According to the

authors, although regularization and noise modelling approaches are effective, they do not necessarily enhance peak performance and may involve a much larger number of iterations than reaching the peak in the original DIP models. Additionally, these approaches rely on comprehensive knowledge of the noise type and level, which is often not available for many applications, resulting in overfitting if the models and hyperparameters are not properly adjusted. The proposed equation is shown as,

$$VAR(t) \approx \frac{1}{W} \sum_{\omega=0}^{W-1} \left| |x_p^{t+\omega} - x_p| \right|^2 \quad (3.5)$$

where

W = window size

x = image pixels

The authors observed that the PSNR curve for image denoising typically follows an inverted U-shaped curve, increasing rapidly at first and then declining due to noise (as seen in Figure 3.6). Conversely, the MSE curve shows the opposite trend. This indicates that the DIP model has a preference towards the important visual content and is capable of learning it more rapidly compared to learning the noise. As a result, the reconstruction quality may reach its highest point before any potential degradation caused by noise.

To achieve superior image quality, it is preferable to locate the peak of the PSNR curve or the valley of the MSE curve. By using Equation (3.5), Wang et al. (2021) found that the variance of the reconstruction sequence decreases as the iteration approached the MSE valley or the PSNR peak. When the iteration reaches the saturation point, where the restored images become close to the original image but with slight variations, early stopping should be applied to avoid overfitting. The calculation of variance involves setting a window size parameter (W) and computing the moving variance. In order to detect the valley more robustly, a patience number (P) is introduced to allow for up to P consecutive steps of variance stagnation (Wang et al., 2021).

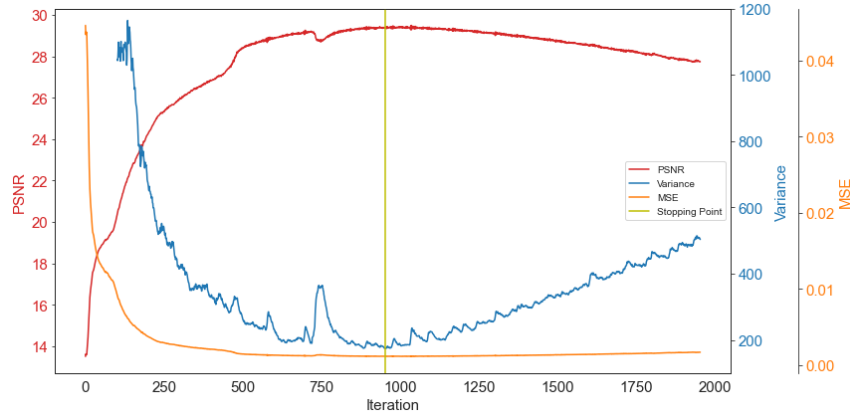


Figure 3.6: Relationship Between MSE, PSNR, and Variance During Image Denoising.

3.8 Proposed Method

Inspired by the variance method, the metric-based early stopping algorithm (MB-ES) has been proposed for identifying the peak and applying early stopping in the image restoration process. This algorithm employs the intermediate performance metric to track the progress and determine the optimal stopping point.

3.8.1 Denoising

A subtype of MB-ES, named denoising-MB-ES, has been developed specifically for denoising task. As an example, the performance of the denoising-MB-ES algorithm was assessed using PSNR method for illustration purposes. Two types of curves were utilized: PSNR_GT, which measures the PSNR between the ground truth and the restored image, and PSNR_INT, which measures the PSNR between intermediate restored images. Figure 3.7 illustrates both curves share similar characteristics. Table 3.4 shows the equation for computing the metrics for each curve, respectively. The function f_{Metric} is utilized for calculating the metrics (PSNR or SSIM) required to plot the curves mentioned earlier. The key difference between Equation (3.6) and Equation (3.7) lies in the choice of reference, with the former using the ground truth and the latter using the intermediate recovered image.

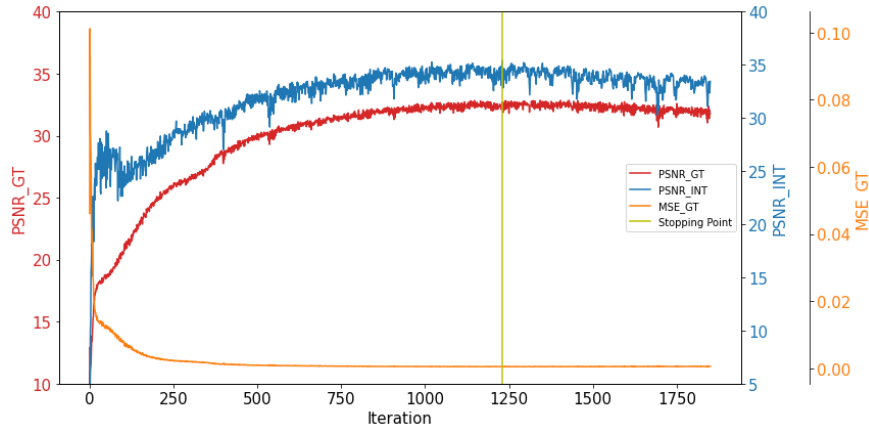


Figure 3.7: Relationship Between MSE_GT, PSNR_GT, and PSNR_INT during Image Denoising. Note: MSE_GT refer to mean squared error metric between restored images and ground truth.

Table 3.4: Metrics Equation for Plotting Curves.

Type of Curve	Equation
PSNR_GT	$f_{Metric}(x_i, z), \quad i \geq 0$ (3.6)
<p>where</p> <p>i = iteration</p> <p>z = ground truth image</p> <p>x = restored image</p>	
PSNR_INT	$f_{Metric}(x_{i-1}, x_i), \quad i \geq 1$ (3.7)

This algorithm incorporates three adjustable parameters that influence its performance. These include:

- (i) Window size multiplier: A variable that is used to determine the length of the window size at each step, resulting in a varying window size.
- (ii) Patience number: Similar to the variance method, this parameter specifies the number of consecutive steps required to terminate the program.
- (iii) Minimum iteration: This parameter establishes when the detection algorithm should start after it has been initialized.

An issue with the denoising-MB-ES algorithm is the possibility that a peak detected in the intermediate performance metric curve may not be the optimal stopping point due to overshooting or severe fluctuation. This can result in premature early stopping and a low-quality image. To address this issue, the density of the surrounding points around the peak is evaluated to determine the desired stopping point. This is accomplished by using varying window sizes, which allows for a more precise detection of the true peak by computing the slope based on the best fit line within the window. The window size is adjusted to different lengths to ensure that the slope is computed over a range of points with respect to the current stopping point and remains fixed until the next peak is discovered. The starting index of the window is then updated to the location of the newly discovered peak, and the slope is computed again with varying window sizes from the updated starting index (as illustrated from Figure B-1 to Figure B-10). Hence, this approach allows for a more robust identification of the true peak of the intermediate performance metric curve and reduces the risk of stopping at an incorrect point due to fluctuations. Figure 3.8 depicts a flowchart illustrating the use of the denoising-MB-ES algorithm for detecting stopping point.

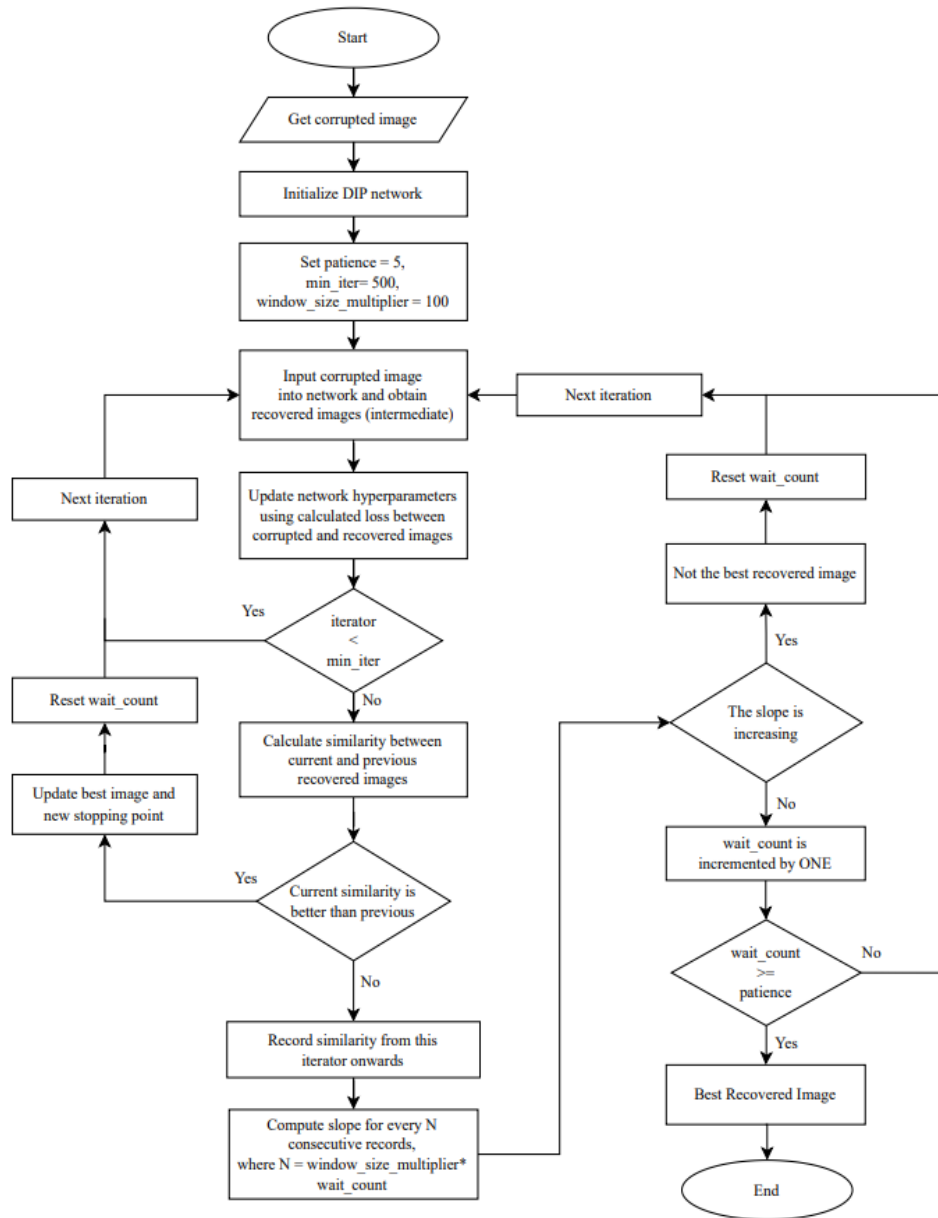


Figure 3.8: Flowchart of the Denoising-MB-ES Algorithm for Early Stopping Detection. Note: The “similarity” could be PSNR or SSIM metrics. The size of the window used to compute the slope of the records is determined by multiplying “wait_count” value with the “window_size_multiplier”. For instance, if the “window_size_multiplier” is set to 100, the window size will be increased for 100 for every “wait_count” steps (e.g., 100, 200, 300, etc.).

A preliminary investigation was conducted to evaluate the effectiveness of the two common metrics used by denoising-MB-ES, namely PSNR and SSIM. The CSet9 images were utilized to compare the performance of these metrics. For each of the nine test images, low and high levels of three different types of noise were applied, with three trials for each noise level, resulting in a total of 162 runs.

For the comparison between the best and detected recovered images, the deviation in terms of PSNR or SSIM can be calculated using the following equation:

$$\% \text{deviation} = \frac{M_D - M_B}{M_B} \times 100 \quad (3.8)$$

where

M_D = the performance metric score based on detected recovered image with respect to ground truth

M_B = the performance metric score based on best recovered image with respect to ground truth

The best recovered image is identified by finding the peak of the performance metric curve between the recovered images and ground truth. In most cases, a negative deviation is obtained, indicating that the algorithm is trying to detect a recovered image with a quality close to the best one. The closer the deviation to 0, the better the performance. However, there may be cases where the SSIM curve slightly behaves differently than the PSNR curve, leading to a positive SSIM deviation, where M_D is better than M_B (see Equation 3.6). This occurs because the SSIM metric considers both the structural information and luminance in the image, which can result in a different ranking of image quality compared to the PSNR metric. Hence, a positive SSIM deviation can be observed when the PSNR method is applied on MB-ES, and the algorithm detects a recovered image with a better SSIM score than the best recovered image. This also applies to PSNR deviation of MB-ES using SSIM method.

Table 3.5 presents the detailed results of the denoising-MB-ES algorithm using the PSNR and SSIM methods for the Lena image. The Stopping Point (SP) refers to the iteration number at which the possible best quality restored image is detected during the restoration process. To calculate the total number of iterations taken by the restoration process, the patience number is also considered. In MB-ES, the patience number specifies additional iterations to determine if the stopping point is the optimal point for the detected image. If there are no other optimal stopping points within the next few iterations specified by the patience number, the program terminates and returns the detected image at the stopping point. During the comparison between PSNR and SSIM methods, the primary focus is not on the number of iterations taken, but rather on the deviations of PSNR and SSIM, as they are the key evaluation criteria for image restoration tasks. The aim is to achieve a smaller deviation value of PSNR and SSIM (closer to zero) in the MB-ES algorithm, which indicates better quality of the recovered image.

Apart from that, for each noise level of the different types of noise, the deviations of PSNR and SSIM for three trials are averaged, resulting in a total of six values, respectively. This averaging process is repeated for all nine images in the CSet9 dataset. As a result, Figure 3.9 and Figure 3.10 provide the summary of the overall findings for all nine CSet9 datasets, with a focus on the comparison between PSNR and SSIM methods, respectively.

Table 3.5: The Proposed Denoising-MB-ES Algorithm Based on PSNR and SSIM Methods on Lena Image.

Image	Noise Type	Noise Level	Trial	PSNR Method								SSIM Method							
				Detected SP			Best SP			Deviation (%)		Detected SP			Best SP			Deviation (%)	
				SP	PSNR	SSIM	SP	PSNR	SSIM	PSNR	SSIM	SP	PSNR	SSIM	SP	PSNR	SSIM	PSNR	SSIM
Lena	Speckle	Low	1	1854	29.31760	0.90192	1873	29.62845	0.90400	-1.04916	-0.23061	1438	29.06267	0.90191	1768	29.4632	0.90450	-1.35945	-0.28582
Lena	Speckle	Low	2	1813	29.71843	0.90703	2006	29.76778	0.90669	-0.16579	0.03803	1598	29.50712	0.90612	1993	29.5475	0.90747	-0.13672	-0.14946
Lena	Speckle	Low	3	1827	29.46919	0.90483	2082	29.65673	0.90287	-0.63237	0.21691	1576	29.29298	0.90311	1848	29.2257	0.90536	0.23020	-0.24839
Lena	Gaussian	Low	1	1518	29.43891	0.90033	1698	29.73361	0.90254	-0.99115	-0.24468	1288	29.28057	0.89855	1636	29.6478	0.90279	-1.23853	-0.46940
Lena	Gaussian	Low	2	1979	29.69000	0.90165	2056	29.72488	0.90092	-0.11733	0.08032	1337	29.07598	0.89623	1897	29.6514	0.90226	-1.94050	-0.66789
Lena	Gaussian	Low	3	1894	29.53778	0.89950	1893	29.68777	0.89998	-0.50520	-0.05392	1111	28.93208	0.89170	1790	29.6231	0.90100	-2.33276	-1.03233
Lena	Shot	Low	1	1288	28.28784	0.88808	1599	28.73250	0.88997	-1.54758	-0.21200	979	27.99916	0.88375	1357	28.4649	0.89088	-1.63610	-0.80041
Lena	Shot	Low	2	1711	28.79860	0.89347	2080	28.80573	0.88889	-0.02474	0.51577	1297	28.45316	0.89043	1717	28.6125	0.89443	-0.55682	-0.44655
Lena	Shot	Low	3	1689	28.64768	0.88714	1509	28.75782	0.89193	-0.38298	-0.53642	1050	28.24166	0.88678	1362	28.6049	0.89266	-1.26973	-0.65784
Lena	Speckle	High	1	863	25.76238	0.86436	1235	26.42617	0.85633	-2.51187	0.93752	201	22.85301	0.79517	876	25.7849	0.86711	-11.37045	-8.29665
Lena	Speckle	High	2	915	25.86054	0.86334	1107	26.30451	0.86238	-1.68781	0.11094	215	23.00021	0.79506	913	25.9717	0.86655	-11.44132	-8.24978
Lena	Speckle	High	3	742	25.67853	0.86350	970	26.23070	0.86479	-2.10508	-0.14975	203	22.51441	0.78948	868	26.0114	0.86842	-13.44404	-9.09003
Lena	Gaussian	High	1	1281	27.52195	0.87070	1263	27.79256	0.87209	-0.97367	-0.15905	223	23.88835	0.80307	1101	27.5722	0.87403	-13.36084	-8.11838
Lena	Gaussian	High	2	917	27.46783	0.87008	1193	27.69750	0.87190	-0.82920	-0.20893	311	24.85774	0.82009	1045	27.4766	0.87372	-9.53130	-6.13887
Lena	Gaussian	High	3	1422	27.59033	0.86486	1278	27.73780	0.86895	-0.53165	-0.47075	374	25.27678	0.83007	1140	27.6580	0.87319	-8.60949	-4.93822
Lena	Shot	High	1	920	26.49954	0.86661	958	26.70231	0.86514	-0.75936	0.16916	235	23.30574	0.80063	920	26.4995	0.86661	-12.05229	-7.61305
Lena	Shot	High	2	887	26.25472	0.86376	1026	26.67056	0.86640	-1.55915	-0.30486	200	23.29146	0.79780	958	26.5034	0.86741	-12.11901	-8.02422
Lena	Shot	High	3	1181	26.32402	0.85456	1049	26.60883	0.86033	-1.07035	-0.67048	207	23.07183	0.79516	868	26.2073	0.86330	-11.96405	-7.89346

Note: SP refer to stopping point. A closer detected stopping point to the best stopping point does not necessarily imply better quality of the recovered image.

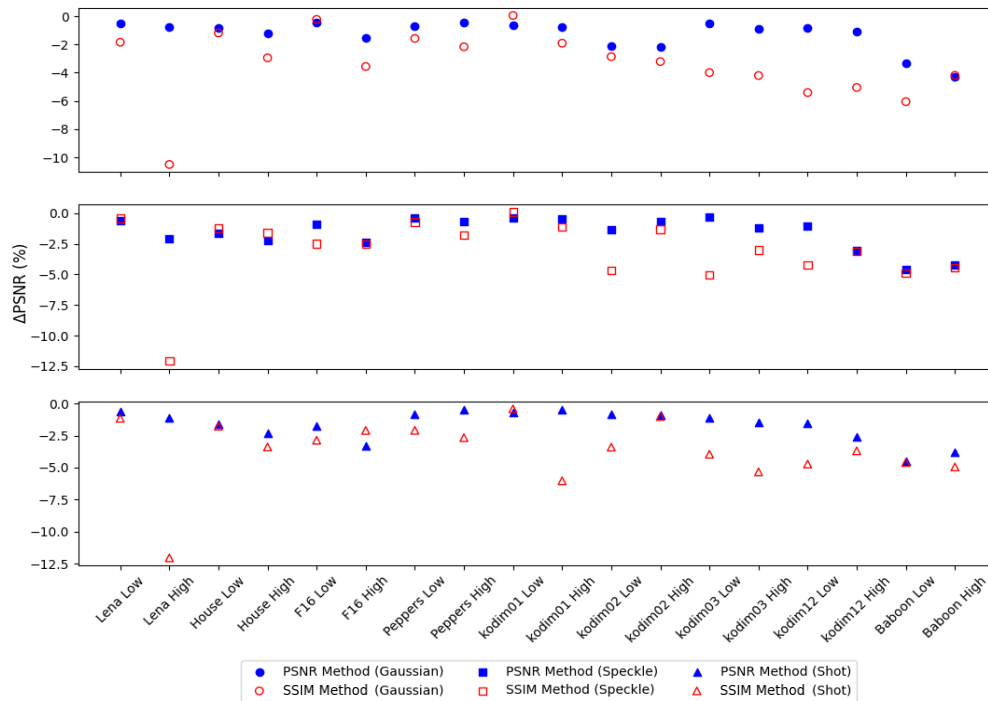


Figure 3.9: Analysis of PSNR Deviation in the Proposed Denoising-MB-ES Algorithm with PSNR and SSIM Methods on CSet9 Images. Refer to Table A-1 for more details.

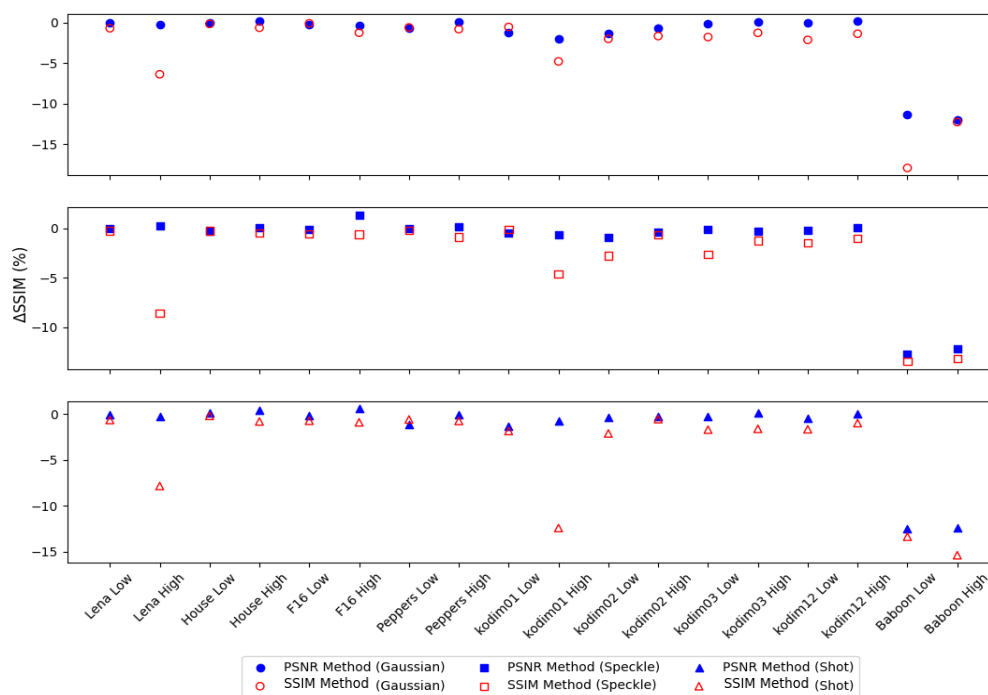


Figure 3.10: Analysis of SSIM Deviation in the Proposed Denoising-MB-ES Algorithm with PSNR and SSIM Methods on CSet9 Images. Refer to Table A-1 for more details.

It can be observed that the PSNR method used in denoising-MB-ES algorithm outperforms the SSIM method. From Table 3.6, The PSNR method shows better performance with a deviation of -1.51030 % and -1.57398 % in PSNR and SSIM, respectively. Therefore, PSNR method is preferable for the denoising-MB-ES algorithm.

Table 3.6: Comparison of PSNR and SSIM Deviations for the Proposed Denoising-MB-ES Using PSNR and SSIM Method.

Metrics	Method	
	PSNR	SSIM
PSNR Deviation (%)	-1.51030	-3.36419
SSIM Deviation (%)	-1.57398	-3.21127

Note: A smaller deviation value (closer to zero) indicates better performance. Please refer Table A-1 for further details.

3.8.2 Super-Resolution

Figure 3.11 shows the comparison of PSNR_INT and PSNR_GT curves for a super-resolution process. Both curves measure the PSNR values obtained at different iterations during the process. As can be seen from the curve, the value of PSNR_INT curve generally increases with the number of iterations, which does not reflect the same behaviour with the PSNR_GT curve. Hence, the PSNR metric is not suitable because the curve does not provide a clear indication of when to stop the super-resolution process. In other words, it is not clear from the curve at which iteration the restored image has reached an acceptable level of quality.

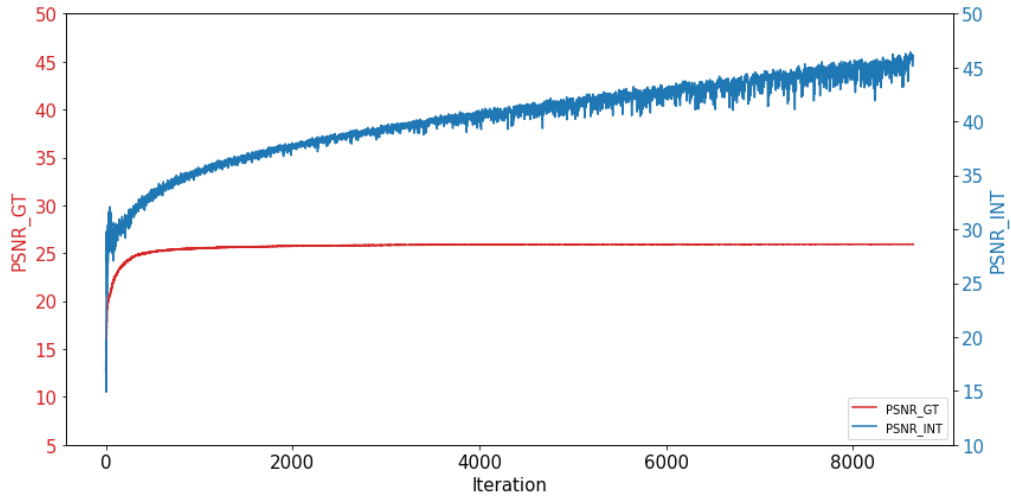


Figure 3.11: Relationship Between PSNR_GT and PSNR_INT During Super-Resolution.

To address this issue, SSIM metric is used instead since it has a fixed range between 0 and 1. Two metrics are utilized, namely SSIM_GT which measures the SSIM between ground truth and recovered HR images, and SSIM_INT which measures the SSIM between intermediate recovered HR images. The SSIM_GT and SSIM_INT are using Equation (3.6) and Equation (3.7), respectively, to plot the curves. Figure 3.12 shows that both SSIM_GT and SSIM_INT curves share a similar trend.

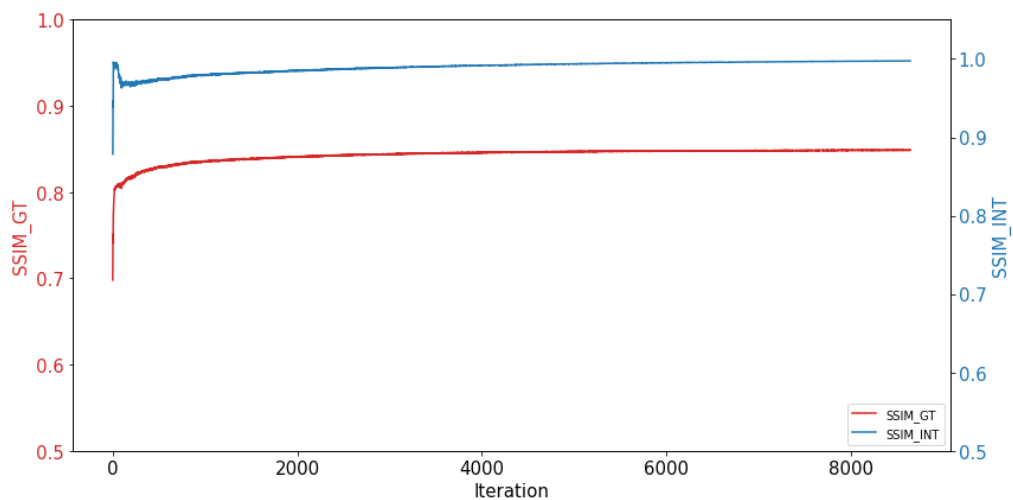


Figure 3.12: Relationship Between SSIM_GT and SSIM_INT During Super-Resolution.

However, the characteristic of the SSIM_INT curve for super-resolution (see Figure 3.12) is distinct from that of the PSNR_INT curve for denoising (see Figure 3.7). The PSNR_INT curve experiences a peak while the SSIM_INT curve does not. Therefore, a subtype of MB-ES, specifically the super-resolution-MB-ES, was developed with slight differences in stopping point detection compared to denoising-MB-ES.

The performance of the super-resolution-MB-ES algorithm can be influenced by three adjustable parameters, which are:

- (i) Patience number: This parameter is similar to the denoising-MB-ES algorithm and specifies the number of consecutive steps required for the program to terminate.
- (ii) Window size: This parameter specifies the number of records stored inside the window for computing the slope.
- (iii) Slope threshold: This parameter is utilized to compare with the slope in order to decide when the process should be terminated.

It has been observed that the SSIM_INT curve initially increases sharply, followed by a gradual increase towards the end, eventually becoming “stable” with almost zero gradient. This characteristic is utilized to determine the optimal stopping point in the super-resolution process. A sliding window approach is employed to compute the slope for the SSIM records that are stored in the window throughout the process. Notably, the sliding window approach is fixed-length, unlike the denoising-MB-ES algorithm where the window size keeps changing.

During the process, if the slope is greater than the threshold, the algorithm updates the best recovered HR image with its corresponding stopping point. The program will terminate once the computed slope is less than the threshold for the desired consecutive number, and lastly it will return the best HR image (as illustrated from Figure B-11 to Figure B-18). Figure 3.13 summarizes the super-resolution-MB-ES algorithm.

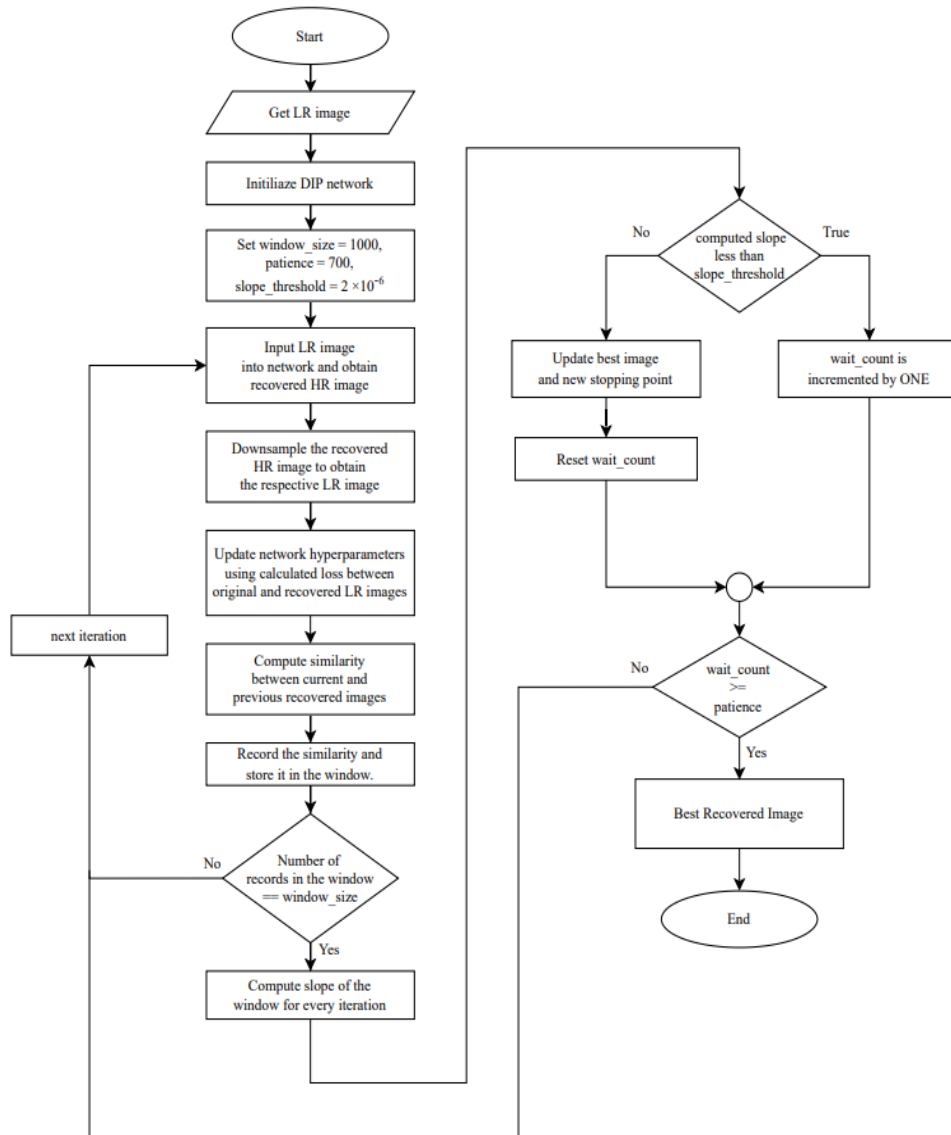


Figure 3.13: Flowchart of the Proposed Super-Resolution-MB-ES Algorithm for Early Stopping Detection. Note: The “similarity” is SSIM metric for super-resolution.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter aims to analyse the performance of the proposed MB-ES algorithm as compared to state-of-the-art methods. Before the benchmarking analyses, investigation was performed on the proposed MB-ES algorithm to fine-tune the parameter values for the optimal performance. The results from the fine-tune process were utilized in the benchmarking analyses, which compared the performance of the MB-ES algorithm with ES-EMV. Subsequently, the denoising-MB-ES and super-resolution-MB-ES algorithms were compared with other existing models. The comparisons are based on the quantitative performance metrics, including PSNR and SSIM, as well as the qualitative visual inspection of the processed images. The results demonstrate the effectiveness and potential of the proposed models for real-world applications. Additionally, the discussion on the benchmarking analyses will provides the insights into the limitations and possible future improvements.

4.2 Investigation of Fine-tuning Parameter Values for the Proposed MB-ES

In this subsection, the fine-tuning process and results for the parameters of the MB-ES algorithm are discussed. The purpose of this investigation is to identify the most suitable values for each parameter to enhance the algorithm's performance in terms of both image quality and time consumption. CSet9 dataset and a variety of parameter values were tested. The primary evaluation criteria focus on the resulting image quality, indicated by the deviation of PSNR and SSIM. The average number of iterations is also taken into consideration as a secondary evaluation criterion, in terms of time consumption.

4.2.1 Denoising-MB-ES Algorithm

Among the three adjustable parameters, only the window size multiplier has the major effect on the performance. To study its impact, the window size multiplier was modified to 50, 100, 200, and 300. The effect of the window size multiplier on the deviations of PSNR and SSIM, as well as the average number of iterations, is illustrated in Figure 4.1.

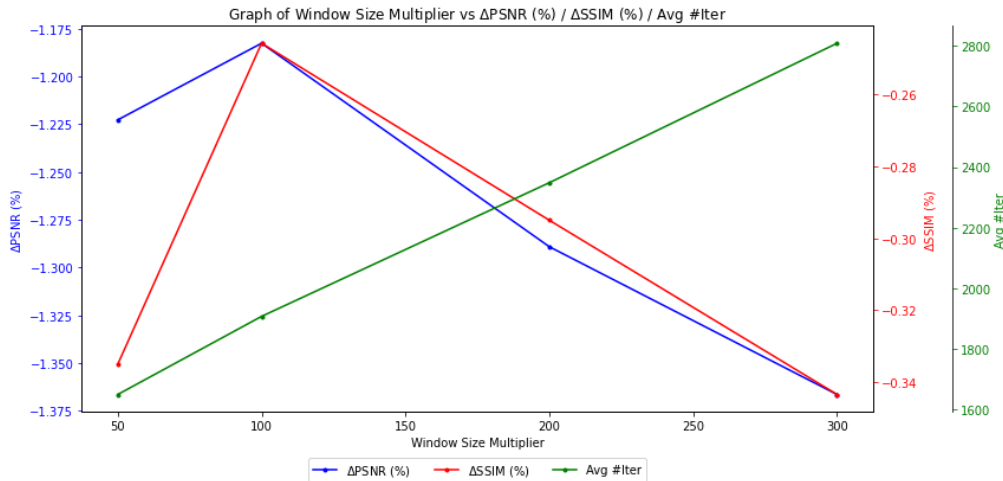


Figure 4.1: Relationship Between Window Size Multiplier and PSNR/SSIM Deviations and Average Number of Iterations.

Hence, it is observed that a window size multiplier of 100 has the lowest deviation in terms of both SSIM and PSNR. Additionally, it provides a balanced trade-off in terms of the average number of iterations required. Therefore, it can be concluded that a window size multiplier of 100 is the optimal parameter value for the denoising-MB-ES algorithm.

4.2.2 Super-Resolution-MB-ES Algorithm

The super-resolution-MB-ES algorithm relies on three parameters: patience, threshold, and window size, to achieve high-quality image restoration. The following subsections describe the steps taken to analyze the relationship between these parameters.

4.2.2.1 Slope Threshold Parameter Tuning

The default parameter values were set to patience = 1000 and window = 1000. Figure 4.2 shows the relationship between the slope threshold parameter and the deviations of PSNR and SSIM, as well as the average number of iterations. Based on the results in Figure 4.2, the threshold parameter is fine-tuned to a value of 2×10^{-6} since it has the lowest deviations in both PSNR and SSIM, which are closer to zero.

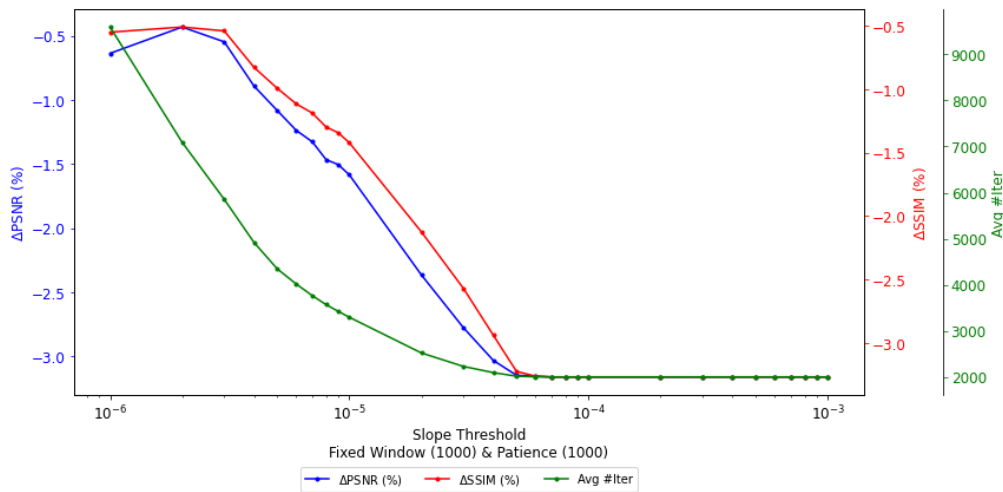


Figure 4.2: Relationship Between Slope Threshold Parameter and PSNR/SSIM Deviations and Average Number of Iterations (with Patience = 1000, Window = 1000).

4.2.2.2 Patience Parameter Tuning

With the threshold parameter fixed at 2×10^{-6} , Figure 4.3 shows the relationship between the patience parameter and the deviations of PSNR and SSIM, as well as the average number of iterations. The results in Figure 4.3 indicate that the optimal value for the patience parameter is 700, as it has the least average number of iterations compared to patience values of 800 and 900.

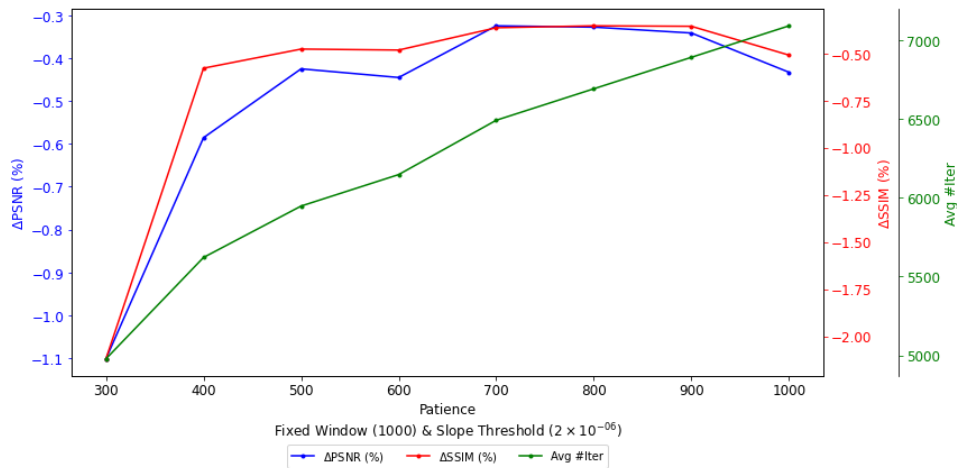


Figure 4.3: Relationship Between Patience Parameter and PSNR/SSIM Deviations and Average Number of Iterations (with Slope Threshold = 2×10^{-6} , Window = 1000).

4.2.2.3 Window Size Tuning

With the threshold parameter fixed at 2×10^{-6} and the patience parameter fixed at 700, Figure 4.4 shows the relationship between the window parameter and the deviations of PSNR and SSIM, as well as the average number of iterations. The results in Figure 4.4 suggest that the optimal window size is 1000, as it has a lower average number of iterations compared to a window size of 800.

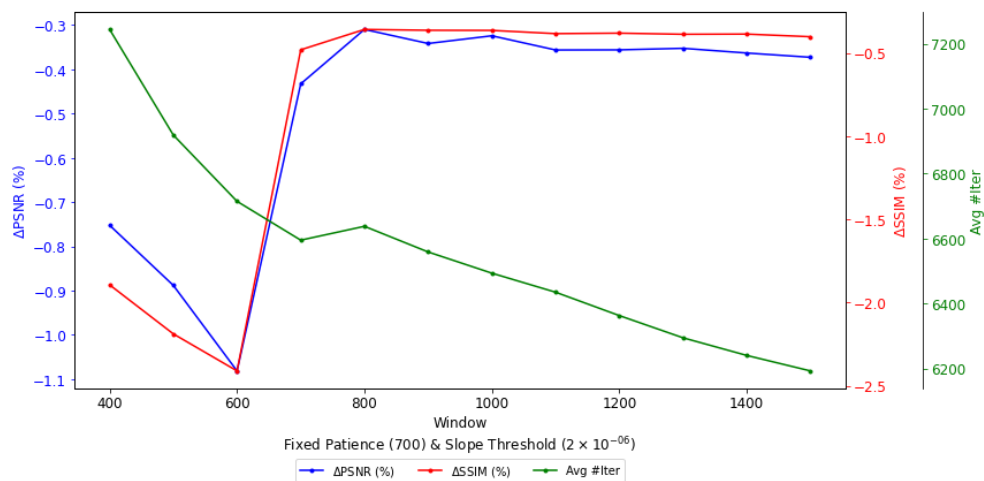


Figure 4.4: Relationship Between Window Parameter and PSNR/SSIM Deviations and Average Number of Iterations (with Slope Threshold = 2×10^{-6} , Patience = 700).

4.2.2.4 Final Fine-Tuning

After the identification of optimal patience and window values as 700 and 1000, respectively, further fine-tuning was done on the threshold parameter. Figure 4.5 shows the relationship between the threshold parameter and the deviations of PSNR and SSIM, as well as the average number of iterations. Based on the results in Figure 4.5, a threshold value of 2×10^{-6} is found to yield the best balance between deviations of PSNR and SSIM, and average number of iterations.

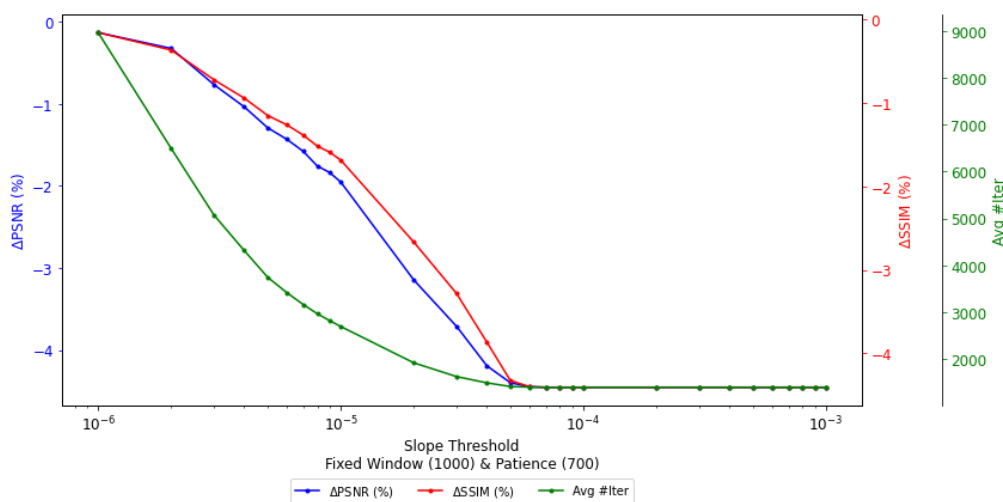


Figure 4.5: Relationship Between Slope Threshold Parameter and PSNR/SSIM Deviations and Average Number of Iterations (with Patience = 700, Window = 1000).

4.3 Comparison of the Proposed MB-ES (PSNR) and ES-EMV (Variance)

To investigate the performance, both models were compared on denoising tasks using DIP model to provide a fair comparison, as the original author implemented the models for this task. The ES-EMV model was tested using its default settings, while MB-ES was configured with the fine-tuned values. A total of 162 runs were conducted, following the similar procedures described in the preliminary investigation of Section 3.8.1. Results are presented in Figure 4.6 and Figure 4.7, which show the PSNR and SSIM deviations of both models on the CSet9 dataset, respectively. A deviation % closer to 0 indicates a closer match to the ground truth. It can be observed that images with different noise

conditions (Gaussian, Speckle and Shot) exhibit diverse restoration results owing to their unique characteristics. Nevertheless, the proposed MB-ES (red) performs better than the ES-EMV (blue) in most of the test images for different noise conditions. Both models excel at reducing low levels of noise as they generally cause less distortion in image quality compared to high levels of noise. However, the proposed MB-ES is better at removing high noise-level images than the ES-EMV.

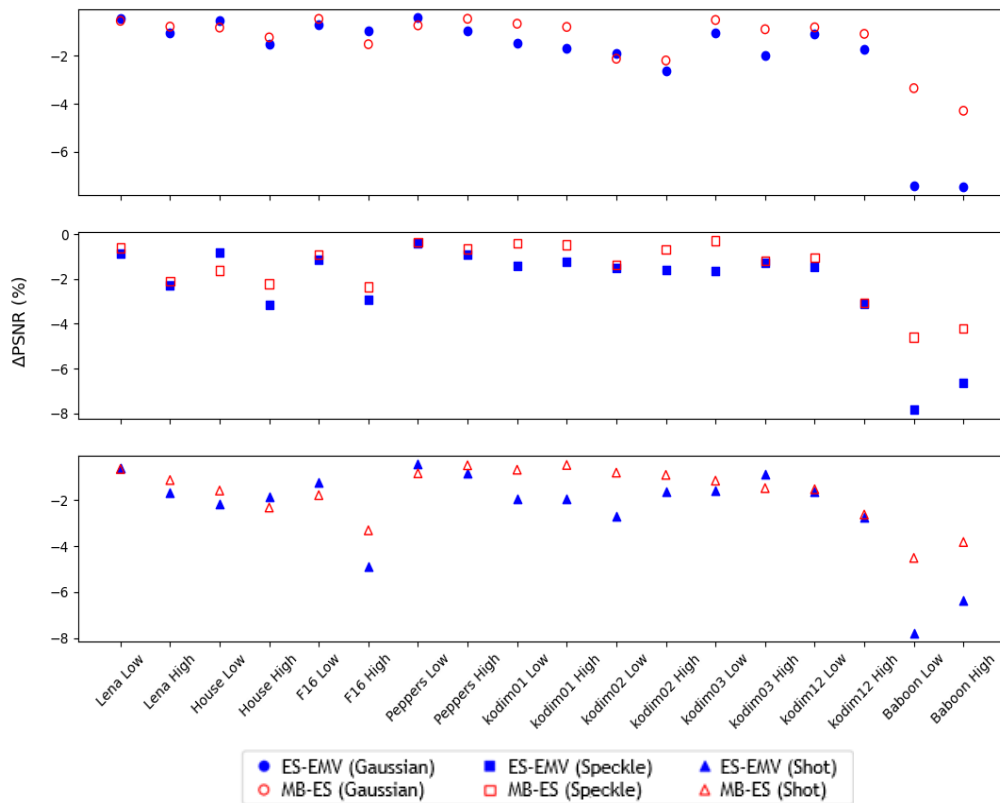


Figure 4.6: Comparison of PSNR Deviation Between the Proposed MB-ES with ES-EMV for the Denoising on CSet9 Images. Refer to Table A-2 for more details.

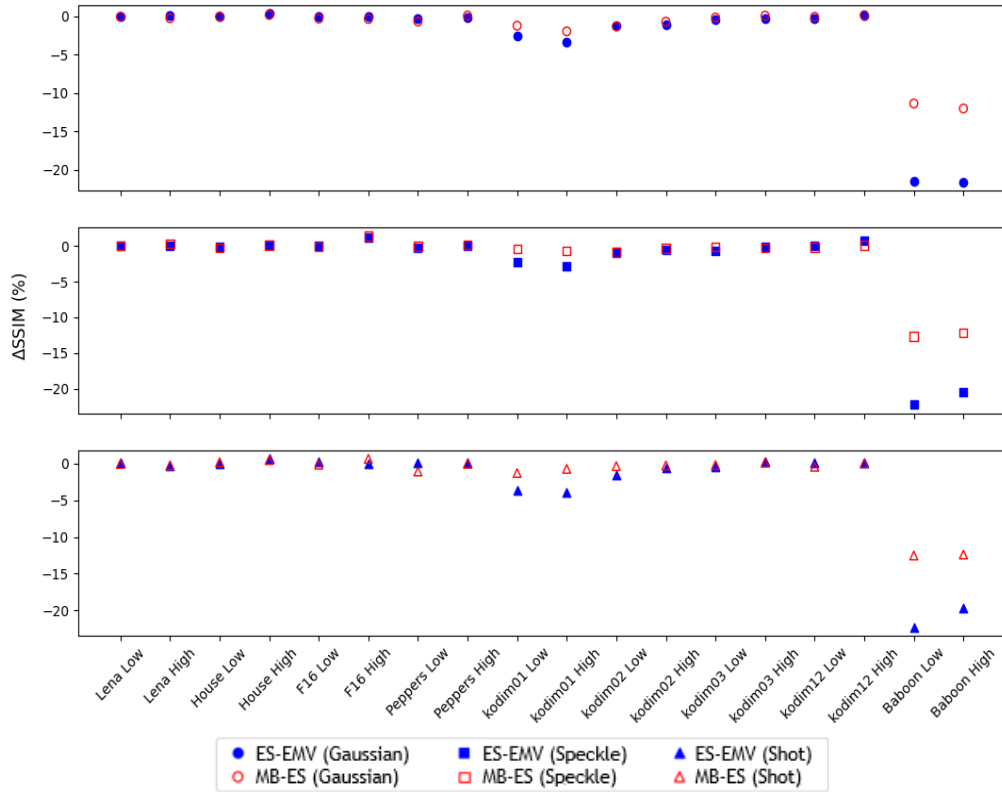


Figure 4.7: Comparison of SSIM Deviation Between the Proposed MB-ES with ES-EMV for the Denoising on CSet9 Images. Refer to Table A-2 for more details.

Table 4.1 summarizes the comparison between the proposed MB-ES and the ES-EMV model. It can be observed that MB-ES is able to deliver better PSNR and SSIM deviations. However, the Baboon images with low- and high-level noises exhibit huge PSNR and SSIM deviations for both models. It might be due to the presence of high-frequency components in images. This issue is to be investigated in future research.

Table 4.1: Comparison of PSNR and SSIM Deviations Between the Proposed MB-ES and ES-EMV.

Metric	Algorithm	
	ES-EMV	MB-ES
PSNR Deviation (%)	-2.18683	-1.51030
SSIM Deviation (%)	-2.80937	-1.57398

Note: A smaller deviation value (closer to zero) indicates better performance. Please refer Table A-2 for further details.

Table 4.2 highlights the comparison between MB-ES and ES-EMV with a focus on their stopping point detection. MB-ES offers several advantages over ES-EMV. Firstly, MB-ES requires fewer iterations ($1813.500 < 2507.685$) to detect the stopping point, resulting in a shorter time for image recovery. Secondly, both MB-ES and ES-EMV require low memory usage. However, MB-ES only considers the performance metric between intermediate images, while ES-EMV utilizes the variance method, which requires dealing with many pixels of the entire image. Lastly, MB-ES has lower design complexity as it only relies on intermediate images, while ES-EMV requires proofing of mathematical formulas and equations to implement the variance method, which demands background knowledge related to the field.

Table 4.2: Comparison of the Proposed MB-ES and ES-EMV.

Comparison Aspect	ES-EMV	MB-ES
Average Number of Iterations	2507.685	1813.500
Memory Usage	Low	Very Low
Design Complexity	High	Low

Note: Please refer to Table A-2 for further details on the average number of iterations.

4.4 Comparison of the Proposed Denoising-MB-ES and Other Denoising Models

To validate the efficiency of the proposed MB-ES in real-world denoising, the algorithm was applied to some related iterative denoising models, which are the deep-decoder, DD (Heckel and Hand, 2018), Stochastic Gradient Langevin Dynamics, SGLD (Cheng et al., 2019) and DIP (Ulyanov et al., 2018). Since the proposed method is implemented based on DIP, it serves as the baseline for comparison. The detected and best stopping points for each model, labelled as “current detection” and “best detection” respectively, are identified from the intermediate PSNR curve (PSNR_INT). The current detection is determined

from the PSNR_INT using the MB-ES algorithm, while the best detection is the recommended stopping point that yields the highest PSNR between the restored image and the ground truth. Typically, the best detection is located at the peak of the PSNR_GT curve. Each iterative denoising models yields different stopping point due to their unique restoration algorithms. To illustrate the performance comparison of denoising models in low- and high-level noise, Figure 4.8 and Figure 4.9 show the denoising results for the F16 image.

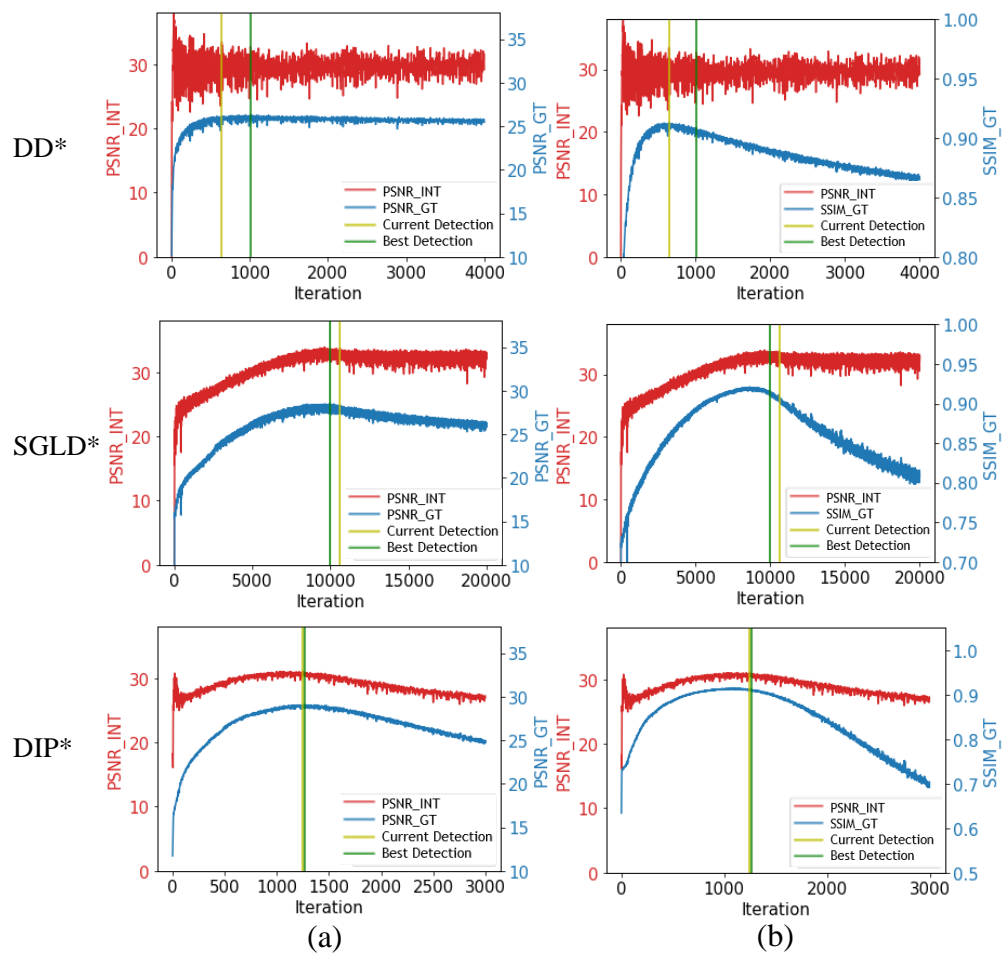


Figure 4.8: Comparison of Low-level Speckle Denoising Results on F16 image for DD*, SGLD* and DIP*: Analysis of PSNR_INT against (a) PSNR_GT and (b) SSIM_GT. Note: The superscript “*” indicates the MB-ES was applied to the model. Further details can be found in Table A-3.

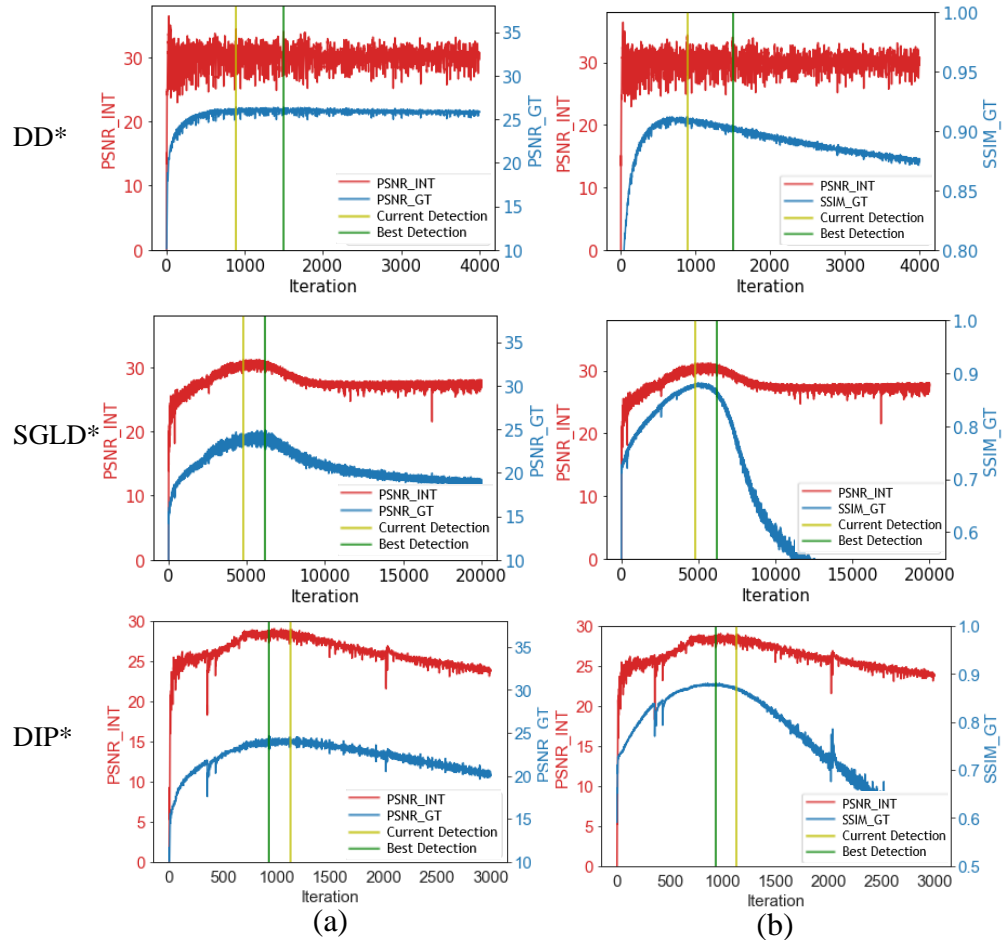


Figure 4.9: Comparison of High-level Speckle Denoising Results on F16 image for DD*, SGLD* and DIP*: Analysis of PSNR_INT Against (a) PSNR_GT and (b) SSIM_GT. Note: The superscript “*” indicates the MB-ES was applied to the model. Further details can be found in Table A-3.

Figure 4.8 and Figure 4.9 demonstrate that the proposed MB-ES algorithm is successful in detecting the PSNR_GT peak during the restoration process. This indicates that the algorithm can effectively identify the best recovered image and achieve a quality close to the ground truth. In addition, PSNR method for MB-ES was able to detect the stopping point that closely corresponds to the SSIM_GT peak. However, minor differences were observed between the best and detected stopping points, which can be considered in future studies. Figure 4.10 shows a visual representation of the denoised images obtained using MB-ES on the discussed denoising models.

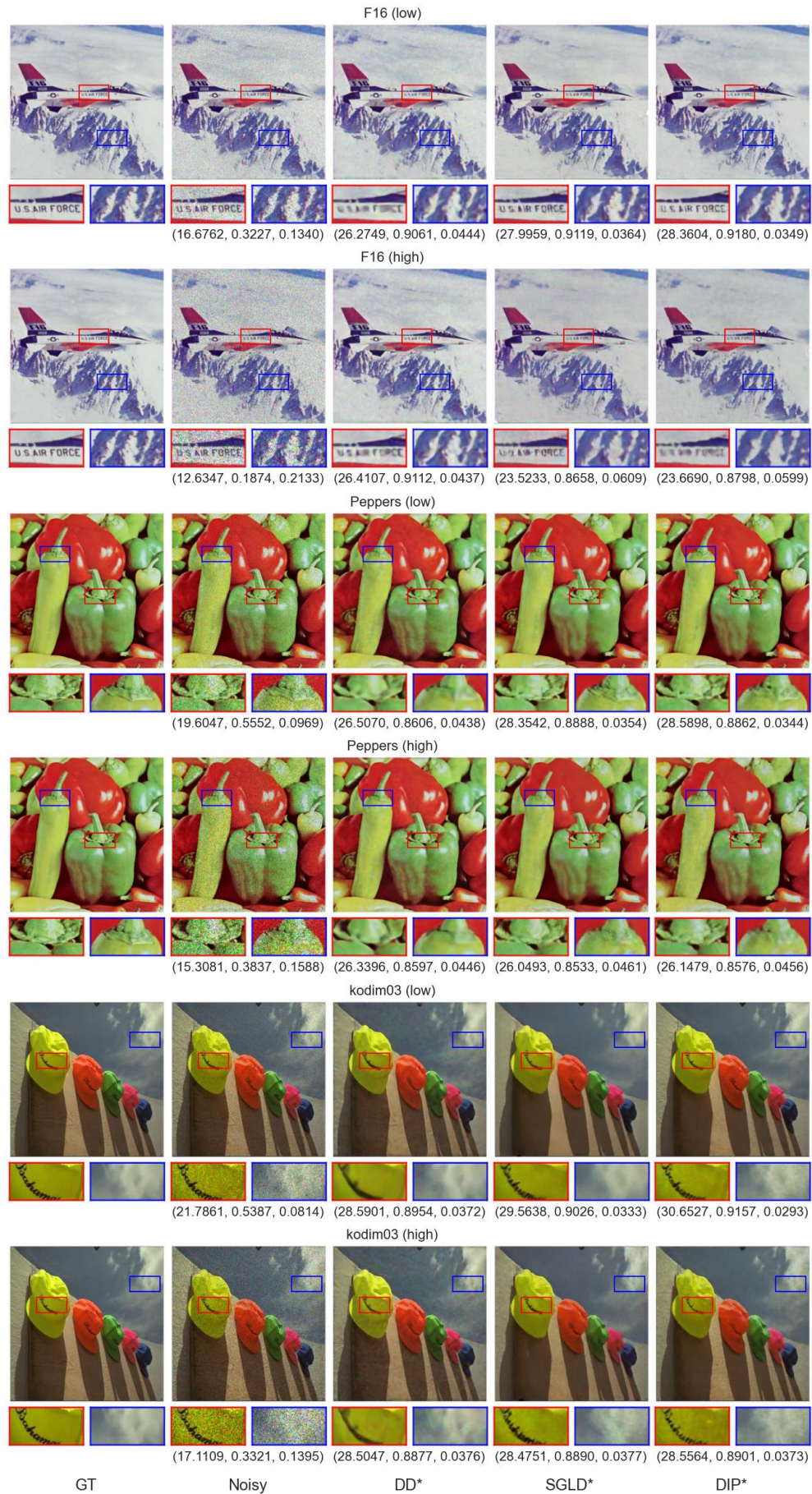


Figure 4.10: Denoising Results of DD*, SGLD* and DIP* on Low- and High-Level Speckle Noises Images: F16, Peppers and kodim03. Note: The superscript “*” indicates the MB-ES was applied to the model. The performance metrics was measured in the form of (PSNR, SSIM, RMSE).

It can be clearly seen that the airplane label of F16 and caps label of kodim03 have been successfully denoised under both low and high noise levels. For Pepper image, the edges of the peppers are effectively preserved while maintaining the details. Hence, the proposed MB-ES is capable of detecting the optimal stopping point for noise removal in iterative restoration models.

4.5 Comparison of the Proposed Super-Resolution-MB-ES and Other Super-Resolution Models

The performance evaluation of super-resolution involved four models: the classical bicubic interpolation method, and two deep learning-based models, SRCNN (Dong et al., 2016), ESPCN (Shi et al., 2016), as well as DIP model (Ulyanov et al., 2018). In order to perform a comprehensive evaluation of their capabilities, the performance of the models was compared using downsampling SF of 2, 3, and 4 on the input images. For instance, an image of size 512×512 was downsampled to sizes of 256×256 (SF = 2), 170×170 (SF = 3), and 128×128 (SF = 4). This allows a thorough analysis of the models' performance across different levels of image resolution, enabling a more accurate assessment of their effectiveness. Figure 4.11 and Figure 4.12 present the results of the comparison between these models in terms of PSNR and SSIM, respectively.

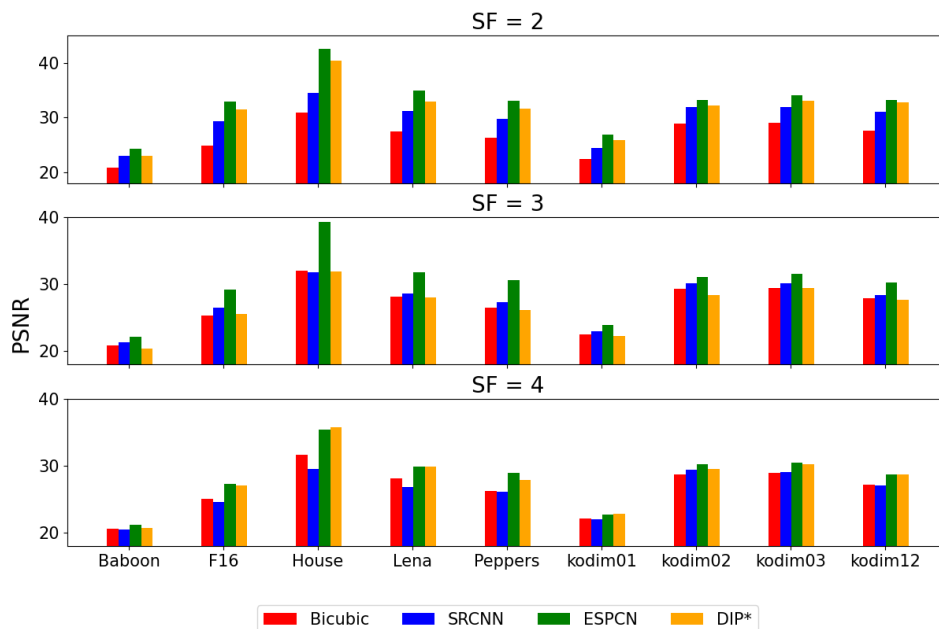


Figure 4.11: Comparison of PSNR for Super-Resolution Models on CSet9 Dataset with Scaling Factors 2, 3, and 4. Note: The superscript “*” indicates the MB-ES was applied to the model. Please refer to Table A-4 for further details.

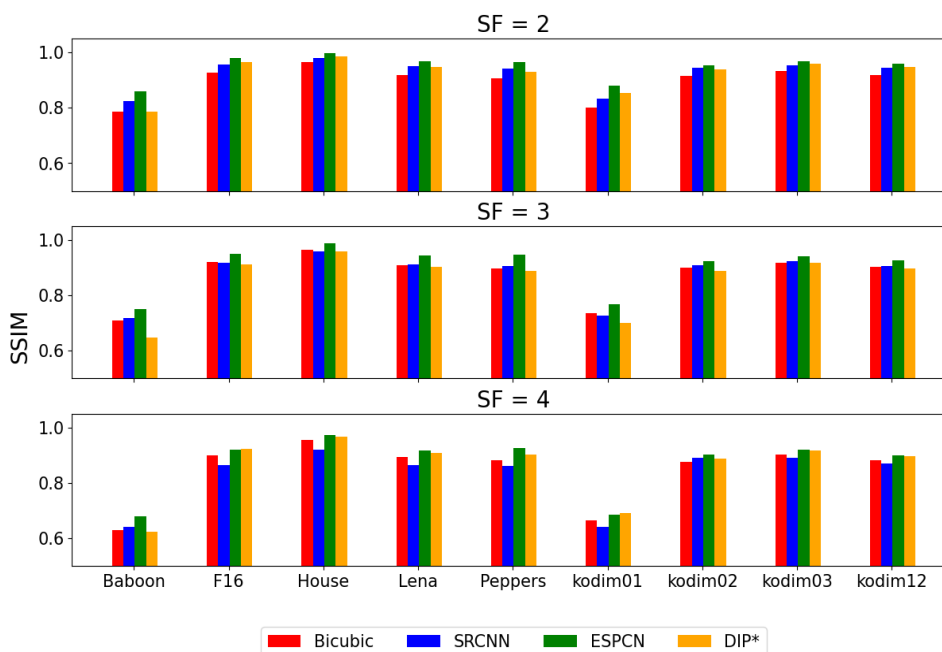


Figure 4.12: Comparison of SSIM for Super-Resolution Models on CSet9 Dataset with Scaling Factors 2, 3, and 4. Note: The superscript “*” indicates the MB-ES was applied to the model. Please refer to Table A-4 for further details.

It can be observed that the DIP model outperforms the classical bicubic method and is on par with the recent deep learning models, SRCNN and ESPCN. It is worth noting that the DIP model has advantages in terms of being an untrained model compared to the trained models, SRCNN and ESPCN. However, there is one exception for the Baboon image where the high-frequency spectrum does not favour the super-resolution algorithm of DIP.

Figure 4.13, Figure 4.14 and Figure 4.15 illustrate the visual representation of the images obtained from multiple models tested for super-resolution at different SF values. It can be clearly seen that the eaves of House image and human eyes of Lena image were successfully recovered for all super-resolution models. However, the image quality was getting reduced as the downsampled SF increases to a value of four, resulting in mosaic effects especially for the bicubic and SRCNN models. Among the super-resolution models, ESPCN has the best visualisation quality due to its unique algorithm and extensive training dataset. On the other hand, the DIP model is able to produce exceptional performance without any mosaic effects on the recovered images. For the Baboon image, the fur edges could not be successfully recovered by any of the models at a scaling factor of 4 due to the inherent nature of the image itself. In general, it can be concluded that DIP using MB-ES is able to still deliver outstanding performance in super-resolution process as compared to the classical and deep-learning based methods.

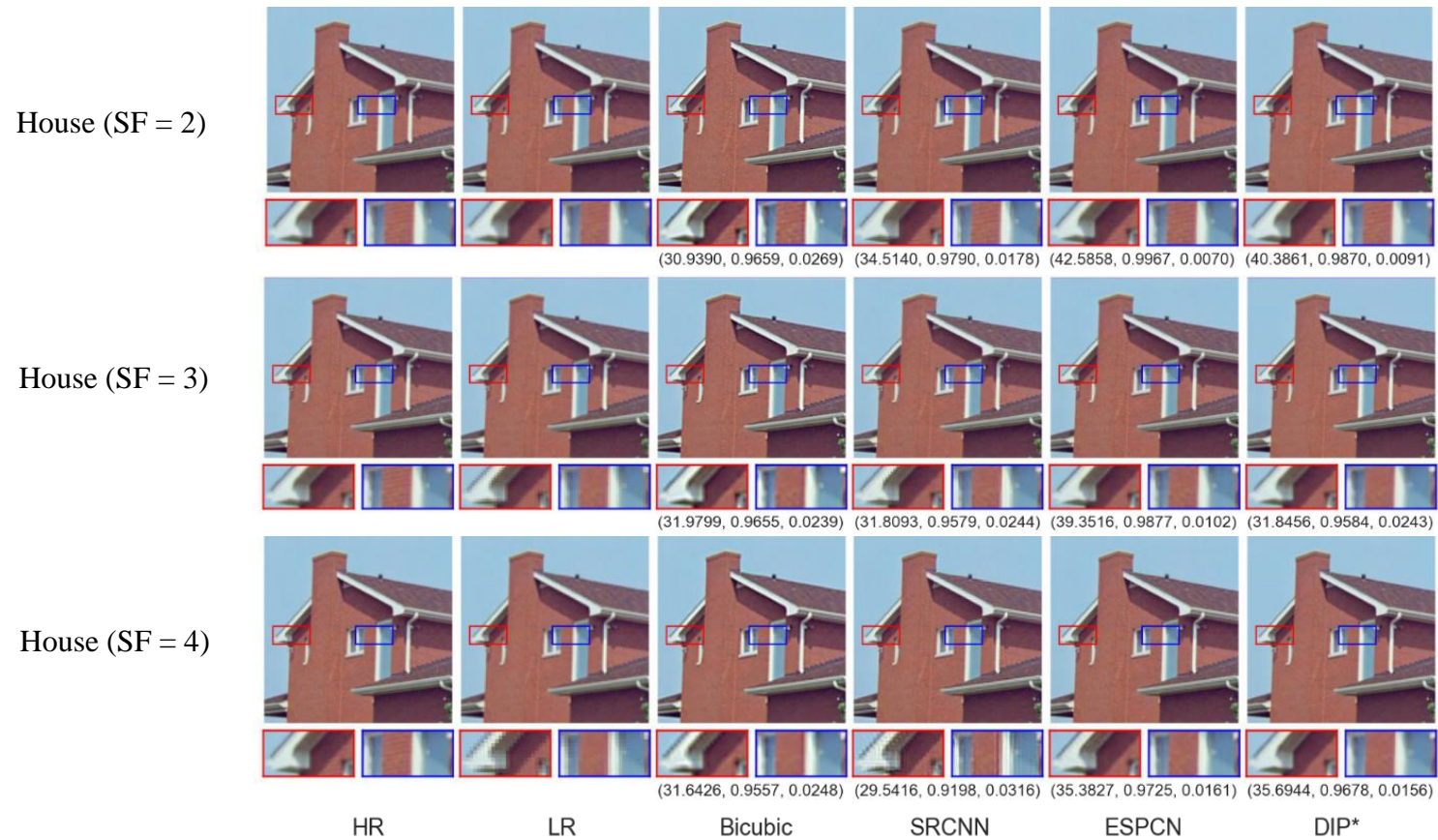


Figure 4.13: Super-Resolution Results of Bicubic, SRCNN, ESPCN and DIP* on Different Scaling Factors of House Image. Note: The superscript “*” indicates the MB-ES was applied to the model. The performance metrics was measured in the form of (PSNR, SSIM, RMSE).

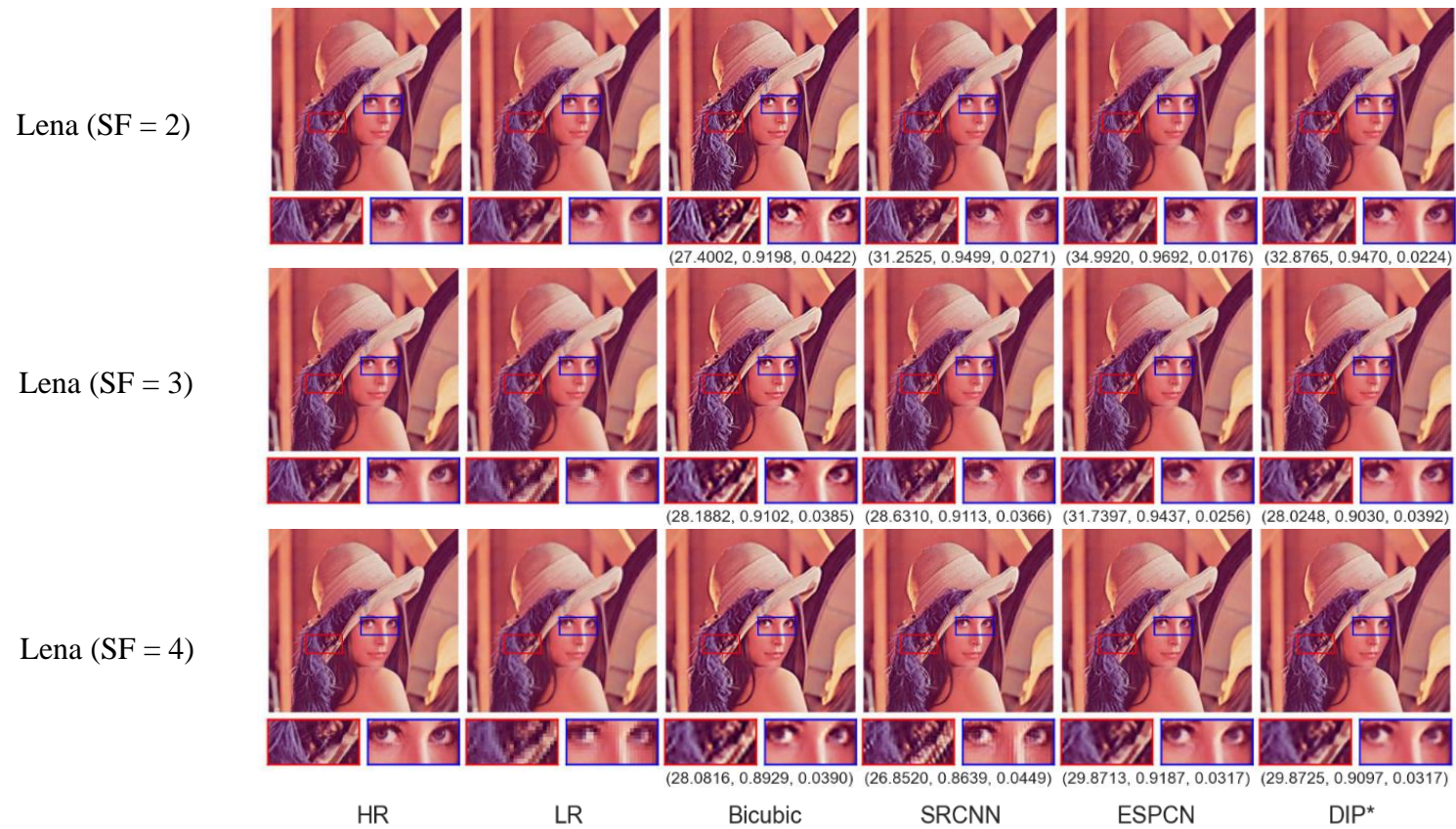


Figure 4.14: Super-Resolution Results of Bicubic, SRCNN, ESPCN and DIP* on Different Scaling Factors of Lena Image. Note: The superscript “*” indicates the MB-ES was applied to the model. The performance metrics was measured in the form of (PSNR, SSIM, RMSE).

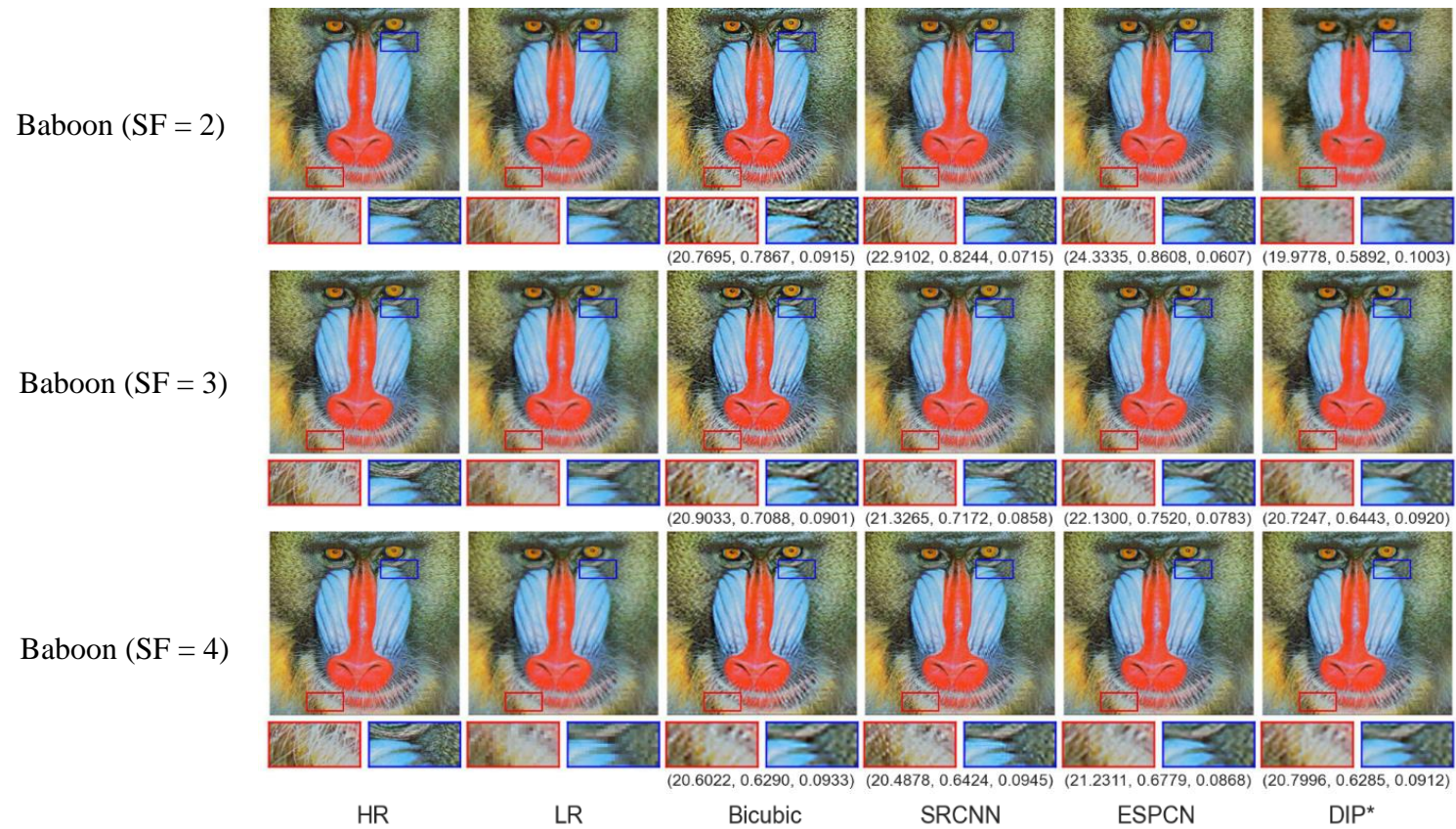


Figure 4.15: Super-Resolution Results of Bicubic, SRCNN, ESPCN and DIP* on Different Scaling Factors of Baboon Image. Note: The superscript “*” indicates the MB-ES was applied to the model. The performance metrics was measured in the form of (PSNR, SSIM, RMSE).

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

Conventional image restoration methods using filters require careful selection of relevant features from individual images. This is not always possible and has become increasingly complex with a greater number of features involved. While deep learning-based methods can address the shortcomings of conventional image restoration methods, the training process is computationally expensive and requires availability of huge datasets. The model training takes considerable time depending on the processing speed of hardware resources. Furthermore, dataset is not always available. Lately, DIP, a learning-free approach to image restoration has emerged as an alternative. DIP uses an untrained convolutional neural network with random initialization. However, DIP requires pre-defined stopping point to recover image, which is impractical in reality. Moreover, different images have their own characteristics that lead to various optimal stopping points. Hence, this study highlights the benefits of using intermediate images to evaluate metrics such as PSNR and SSIM in image restoration, which are applied to the DIP method.

The proposed MB-ES method provides better performance than the ES-EMV method in terms of the average number of iterations taken to detect the optimal stopping point and return the best quality image. Apart from that, the proposed MB-ES algorithm can be applied on the recent iterative denoising model, and is proven to be effective in tracking the peak of the curve between restored images and ground truth, resulting in obtaining the best quality image. Furthermore, the proposed MB-ES algorithm for super-resolution tasks outperforms the classical bicubic method and shows comparable performance with recent deep learning super-resolution models. It is worth mentioning that the DIP model does not require pre-training on massive datasets and still delivers decent performance on image restoration tasks.

5.2 Recommendations for future work

In terms of future work, there are several areas that can be explored to further improve the effectiveness of the proposed MB-ES approach on the DIP model. Besides, the capabilities of the DIP architecture in restoring images can be further enhanced. Some potential future works include:

- (i) The DIP architecture can be modified to enhance the behaviour of the intermediate images curve for efficient detection of the best quality image. These modifications might include the selection of the network, number of layers, loss function, optimizer, and other relevant hyperparameters.
- (ii) Additionally, alternative metrics beyond PSNR or SSIM could be implemented to evaluate intermediate images such as a blend of PSNR and SSIM, or other metrics. These modifications potentially improve image restoration outcomes for various tasks beyond those examined in this study, such as deblurring or inpainting, and provide further insights into the effectiveness of the approach for different types of image degradation.
- (iii) To overcome the limitation of dataset selection, a more diverse range of datasets can be considered for evaluation.

REFERENCES

- A.M, R., W.M, K., M.A, E. and Aoud, M., 2014. Fast NAS-RIF Algorithm Using Iterative Conjugate Gradient Method. *Signal & Image Processing: An International Journal*, 5(2), pp.63–72.
- Boyat, A.K. and Joshi, B.K., 2015. A Review Paper: Noise Models in Digital Image Processing. *Signal & Image Processing an International Journal*, [e-journal] 6(2), pp.64–75. <http://doi.org/10.5121/sipij.2015.6206>.
- Cheng, Z., Gadelha, M., Maji, S. and Sheldon, D., 2019. A Bayesian Perspective on the Deep Image Prior. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, [e-journal], pp.5438-5446. <https://doi.org/10.1109/CVPR.2019.00559>.
- Das, Sanjib, Saikia, J., Das, Soumita and Goñi, N., 2015. A COMPARATIVE STUDY OF DIFFERENT NOISE FILTERING TECHNIQUES IN DIGITAL IMAGES. *International Journal of Engineering Research and General Science*, 3(5), pp.180–191.
- Dong, C., Loy, C.C., He, K. and Tang, X., 2016. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [e-journal] 38(2), pp.295-307. <https://doi.org/10.1109/TPAMI.2015.2439281>.
- Eigen, D., Puhrsch, C. and Fergus, R., 2014. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network, [online] Available at: <<http://arxiv.org/abs/1406.2283>> [Accessed: 31 August 2022].
- Fisher, R., Perkins, S., Walker, A. and Wolfart, E., 2003, *Image Processing Learning Resources: Conservative Smoothing*. [online] Available at: <[https://homepages.inf.ed.ac.uk/rbf/HIPR2/csmooth.htm#:~:text=Conservative%20smoothing%20is%20a%20noise,sharp%20edges\)%20in%20an%20image.>](https://homepages.inf.ed.ac.uk/rbf/HIPR2/csmooth.htm#:~:text=Conservative%20smoothing%20is%20a%20noise,sharp%20edges)%20in%20an%20image.>) [Accessed: 31 August 2022].
- Gholizadeh-Ansari, M., Alirezaie, J. and Babyn, P., 2020. Deep Learning for Low-Dose CT Denoising. *Journal of Digital Imaging*, [e-journal] 33(2), pg.504-515. <https://doi.org/10.1007/s10278-019-00274-4>.
- Great Learning Team, 2020, *Introduction to Image Pre-processing / What is Image Pre-processing?*. [online] Available at: <<https://www.mygreatlearning.com/blog/introduction-to-image-pre-processing/>> [Accessed: 5 September 2022].
- Heckel, R. and Hand, P., 2018. Deep Decoder: Concise Image Representations from Untrained Non-convolutional Networks, [online] Available at: <<https://arxiv.org/abs/1810.03982>> [Accessed: 28 March 2023].

- Hendrycks, D. and Dietterich, T., 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, [online] Available at: <<https://arxiv.org/abs/1903.12261>> [Accessed: 5 September 2022].
- Isogawa, K., Ida, T., Shiodera, T. and Takeguchi, T., 2018. Deep Shrinkage Convolutional Neural Network for Adaptive Noise Reduction. *IEEE Signal Processing Letters*, 25(2), pp.224–228. <https://doi.org/10.1109/LSP.2017.2782270>.
- Jiang, M., 2006. Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods. *BioMedical Engineering OnLine*, [e-journal] 5(1), p.38. <https://doi.org/10.1186/1475-925X-5-38>.
- Kanrar, S., and Maji, S., 2022. A Study on Image Restoration and Analysis. In: Kumar, N., Shahnaz, C., Kumar, K., Abed Mohammed, M., Raw, R.S., eds. 2022. *Advance Concepts of Image Processing and Pattern Recognition. Transactions on Computer Systems and Networks*. [e-book] Springer, Singapore. https://doi.org/10.1007/978-981-16-9324-3_3.
- Khare, C., Nagwanshi, K. and Nagwanshi, 2011. Implementation and Analysis of Image Restoration Techniques. *International Journal of Computer Trends and Technology-May to June Issue 2011*, [e-journal] 54, pp.1–6. <http://ijcttjournal.org/Volume1/issue-2/ijcttjournal-v1i2p12.pdf>.
- Kim, J., Lee, J.K. and Lee, K.M., 2016a. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, [e-journal] pp.1646-1654. <https://doi.org/10.1109/CVPR.2016.182>.
- Kim, J., Lee, J.K. and Lee, K.M., 2016b. Deeply-Recursive Convolutional Network for Image Super-Resolution. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, [e-journal] pp. 1637-1645. <https://doi.org/10.1109/CVPR.2016.181>.
- Kitchener, M.A., 2012. *Investigations into image restoration*. PhD thesis, University of Wollongong, [online] Available at: <<https://ro.uow.edu.au/theses/3900/>> [Accessed: 8 September 2022].
- Kupyn, O. et al., 2018. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, [e-journal] pp. 8183-8192. <https://doi.org/10.1109/CVPR.2018.00854>.
- Kupyn, O., Martyniuk, T., Wu, J. and Wang, Z., 2019. DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, [e-journal] pp. 8877-8886. <https://doi.org/10.1109/ICCV.2019.00897>.

- Lu, H. et al., 2017. Underwater Optical Image Processing: A Comprehensive Review, [online] Available at: <<https://arxiv.org/abs/1702.03600>> [Accessed: 8 September 2022].
- Madam Nimisha, T., Sunil, K. and Rajagopalan, A.N., 2018. Unsupervised Class-Specific Deblurring. *Proceedings of the European Conference on Computer Vision (ECCV)*, [online] pp.353–369. Available at: <<http://www.ee.iitm.ac.in/ipcvlab/>> [Accessed: 2 September 2022].
- Mahony, N.O. et al., 2019. Deep Learning vs. Traditional Computer Vision, [online] Available at: <<http://arxiv.org/abs/1910.13796>> [Accessed: 2 September 2022].
- Nah, S., Kim, T.H. and Lee, K.M., 2017. Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, [e-journal] pp.257-265. <https://doi.org/10.1109/CVPR.2017.35>.
- Navaneethakrishnan, R., 2014. A Comparative Study and Analysis of Image Restoration Techniques Using Different Images Formats. *International Journal of Advanced Research*, 1(3), pp.131–137.
- Rani, S., Jindal, S. and Kaur, B., 2016. A Brief Review on Image Restoration Techniques. *International Journal of Computer Applications*, [e-journal] 150(12), pp.30-33. <http://doi.org/10.5120/ijca2016911623>.
- Richardson, W.H., 1972. Bayesian-Based Iterative Method of Image Restoration. *Journal of the Optical Society of America*, 62, pp.55–59.
- Sharma, Swati, Sharma, Shipra and Mehra, R., 2013. Image Restoration using Modified Lucy Richardson Algorithm in the Presence of Gaussian and Motion Blur. *Research India Publications*, [online] 3(8), pp.1063–1070. Available at: <https://www.ripublication.com/aeee/056_pp%20%20%20%201063-1070.pdf> [Accessed: 2 September 2022].
- Shi, W. et al., 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, [e-journal] pp.1874-1883. <https://doi.org/10.1109/CVPR.2016.207>.
- Shimamura, Tetsuya, Furuya, H. and Eda, S., 2009. Image restoration via Wiener filtering with improved noise estimation, [online] <Available at: <https://www.researchgate.net/publication/229052293>> [Accessed: 2 September 2022].
- Shocher, A., Cohen, N. and Irani, M., 2018. Zero-Shot Super-Resolution using Deep Internal Learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, [e-journal] pp.3118-3126. <https://doi.org/10.1109/CVPR.2018.00329>.

Su, J., Xu, B. and Yin, H., 2022. A survey of deep learning approaches to image restoration. *Neurocomputing*, [e-journal] 487(1), pp.46–65. <https://doi.org/10.1016/j.neucom.2022.02.046>.

Tai, Y., Yang, J., Liu, X. and Xu, C., 2017. MemNet: A Persistent Memory Network for Image Restoration. *2017 IEEE International Conference on Computer Vision (ICCV)*, [e-journal] pp.4549-4557. <https://doi.org/10.1109/ICCV.2017.486>.

Ulyanov, D., Vedaldi, A. and Lempitsky, V., 2018. Deep Image Prior. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, [e-journal] pp.9446-9454. <https://doi.org/10.1109/CVPR.2018.00984>.

Wang, H. et al., 2021. Early Stopping for Deep Image Prior, [online] Available at: <<http://arxiv.org/abs/2112.06074>> [Accessed: 28 March 2023].

Yadav, S., Jain, C. and Chugh, A., 2016. Evaluation of Image Deblurring Techniques. *International Journal of Computer Applications*, [e-journal] 139(12), pp.32-36. <http://doi.org/10.5120/ijca2016909492>.

Ye, J.C., Han, Y. and Cha, E., 2018. Deep Convolutional Framelets: A General Deep Learning Framework for Inverse Problems. *SIAM Journal on Imaging Sciences*, [e-journal] 11(2), pp.991-1048. <https://doi.org/10.1137/17M1141771>.

Zhang, H., Dai, Y., Li, H. and Koniusz, P., 2019. Deep Stacked Hierarchical Multi-Patch Network for Image Deblurring. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, [e-journal] pp.5971-5979. <https://doi.org/10.1109/CVPR.2019.00613>.

Zhang, K. et al., 2017. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, [e-journal] 26(7), pp.3142-3155. <https://doi.org/10.1109/TIP.2017.2662206>.

Zhang, K., Zuo, W. and Zhang, L., 2018. FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising. *IEEE Transactions on Image Processing*, [e-journal] 27(9), pp.4608-4622. <https://doi.org/10.1109/TIP.2018.2839891>.

Zheng, Y., 1989. A Fast Image Deblurring Algorithm Using the Wiener Filter and the Hartley Transform. In: Thompson Donald O. and Chimenti, D.E., eds. 1989. *Review of Progress in Quantitative Nondestructive Evaluation: Volume 8, Part A and B*. Springer US, Boston, MA, pp. 735–742.

Zhu, Y. and Huang, C., 2012. An Improved Median Filtering Algorithm for Image Noise Reduction. *Physics Procedia*, [e-journal] 25, pp.609–616. <http://doi.org/10.1016/j.phpro.2012.03.133>.

APPENDICES

Appendix A: Additional Tables

Table A-1: Overall Comparison of Denoising-MB-ES Algorithm Performance on CSet9 Images based on PSNR and SSIM Methods. The best score is highlighted in red.

Image	Noise Level	Noise Type	PSNR Method		SSIM Method	
			Deviation (%)		Deviation (%)	
			PSNR	SSIM	PSNR	SSIM
Lena	Low	Speckle	-0.61577	0.00811	-0.42199	-0.22789
		Gaussian	-0.53790	-0.07276	-1.83726	-0.72321
		Shot	-0.65177	-0.07755	-1.15422	-0.63493
	High	Speckle	-2.10159	0.29957	-12.08527	-8.54549
		Gaussian	-0.77817	-0.27958	-10.50054	-6.39849
		Shot	-1.12962	-0.26873	-12.04512	-7.84358
House	Low	Speckle	-1.62460	-0.26190	-1.22761	-0.24192
		Gaussian	-0.81667	-0.05310	-1.18159	-0.18664
		Shot	-1.58575	0.16139	-1.77645	-0.18445
	High	Speckle	-2.21733	0.09623	-1.61099	-0.47667
		Gaussian	-1.22780	0.14431	-2.94424	-0.65962
		Shot	-2.32439	0.44584	-3.39373	-0.79523
F16	Low	Speckle	-0.91919	-0.07680	-2.49951	-0.56049
		Gaussian	-0.45661	-0.32679	-0.22181	-0.11658
		Shot	-1.78424	-0.16707	-2.86503	-0.71871
	High	Speckle	-2.36519	1.38238	-2.54406	-0.59867
		Gaussian	-1.51400	-0.37720	-3.56199	-1.27238
		Shot	-3.3143	0.61851	-2.10446	-0.88788
Peppers	Low	Speckle	-0.37862	-0.02120	-0.71649	-0.20382
		Gaussian	-0.72568	-0.70207	-1.56851	-0.61950
		Shot	-0.83702	-1.10534	-2.09640	-0.57826
	High	Speckle	-0.65834	0.18280	-1.80747	-0.85450
		Gaussian	-0.45283	0.07510	-2.16237	-0.83168
		Shot	-0.48751	-0.05531	-2.66673	-0.74319
kodim01	Low	Speckle	-0.41704	-0.43217	0.08253	-0.08308
		Gaussian	-0.65694	-1.24820	0.05696	-0.55976
		Shot	-0.68046	-1.30495	-0.41853	-1.83928
	High	Speckle	-0.47369	-0.64682	-1.12875	-4.61360
		Gaussian	-0.79105	-1.97890	-1.90475	-4.81007
		Shot	-0.47435	-0.74474	-6.03004	-12.43427
kodim02	Low	Speckle	-1.36064	-0.90849	-4.66236	-2.76410
		Gaussian	-2.11769	-1.33488	-2.86159	-2.02938
		Shot	-0.80612	-0.37096	-3.40882	-2.10258

	High	Speckle	-0.68677	-0.33656	-1.30691	-0.60549
		Gaussian	-2.19071	-0.72003	-3.20912	-1.68206
		Shot	-0.90468	-0.28160	-1.01308	-0.54103
kodim03	Low	Speckle	-0.30322	-0.07469	-5.03724	-2.59298
		Gaussian	-0.50200	-0.18108	-3.99036	-1.80705
		Shot	-1.15672	-0.22841	-3.96045	-1.69144
	High	Speckle	-1.20334	-0.24147	-3.01072	-1.23371
		Gaussian	-0.88843	0.05729	-4.19772	-1.28846
		Shot	-1.47964	0.10834	-5.33545	-1.59937
kodim12	Low	Speckle	-1.06630	-0.17939	-4.22734	-1.43718
		Gaussian	-0.80805	-0.10259	-5.40409	-2.15940
		Shot	-1.52984	-0.45597	-4.72892	-1.64616
	High	Speckle	-3.05227	0.12072	-3.12247	-0.96943
		Gaussian	-1.08043	0.12643	-5.04509	-1.39661
		Shot	-2.63025	-0.00409	-3.69749	-0.97397
Baboon	Low	Speckle	-4.60521	-12.68824	-4.87232	-13.46522
		Gaussian	-3.34769	-11.37382	-6.04949	-17.94385
		Shot	-4.51421	-12.52342	-4.60517	-13.38317
	High	Speckle	-4.21210	-12.20148	-4.44957	-13.15723
		Gaussian	-4.28815	-12.02414	-4.18284	-12.29107
		Shot	-3.82350	-12.38953	-4.95113	-15.40369
Average			-1.51030	-1.57398	-3.36419	-3.21127

Table A-2: Overall Comparison of Denoising Performance of Proposed MB-ES and ES-EMV on CSet9 Images. The best score is highlighted in red.

Image	Noise Level	Noise Type	ES-EMV			MB-ES		
			Deviation (%)		SP	Deviation (%)		SP
			PSNR	SSIM		PSNR	SSIM	
Lena	Low	Speckle	-0.87157	-0.02125	1858	-0.61577	0.00811	1827
		Gaussian	-0.45601	-0.04834	1734	-0.5379	-0.07276	1825
		Shot	-0.61949	0.03390	1501	-0.65177	-0.07755	1570
	High	Speckle	-2.30761	-0.00273	892	-2.10159	0.29957	898
		Gaussian	-1.02203	0.03636	1053	-0.77817	-0.27958	1038
		Shot	-1.66766	-0.29522	877	-1.12962	-0.26873	898
House	Low	Speckle	-0.81472	-0.11774	1472	-1.6246	-0.26190	1680
		Gaussian	-0.52396	0.01309	1314	-0.81667	-0.05310	1213
		Shot	-2.16771	-0.01989	1419	-1.58575	0.16139	1016
	High	Speckle	-3.13974	0.21071	664	-2.21733	0.06713	751
		Gaussian	-1.52369	0.32322	984	-1.2278	0.17341	859
		Shot	-1.87765	0.64487	742	-2.32439	0.44584	549
F16	Low	Speckle	-1.14151	-0.02570	1917	-0.91919	-0.07680	1770
		Gaussian	-0.68620	-0.07301	2046	-0.45661	-0.32679	2283
		Shot	-1.24433	0.20394	1632	-1.78424	-0.16707	1736
	High	Speckle	-2.94088	1.18013	873	-2.36519	1.38238	922
		Gaussian	-0.94180	-0.02140	1211	-1.51400	-0.37720	1489
		Shot	-4.90841	0.00706	916	-3.31430	0.61851	948
Peppers	Low	Speckle	-0.40574	-0.20862	3403	-0.37862	-0.02120	1930
		Gaussian	-0.40646	-0.25171	3576	-0.72568	-0.70207	2542
		Shot	-0.42921	0.04092	2512	-0.83702	-1.10534	2118
	High	Speckle	-0.89608	0.07516	1681	-0.65834	0.18280	1176
		Gaussian	-0.95648	-0.21403	2098	-0.45283	0.07510	1242
		Shot	-0.83620	0.13427	1801	-0.48751	-0.05531	1225
kodim01	Low	Speckle	-1.43642	-2.22123	3423	-0.41704	-0.43217	2929
		Gaussian	-1.48776	-2.55615	2782	-0.65694	-1.24820	1945
		Shot	-1.93326	-3.65193	2140	-0.68046	-1.30495	1520
	High	Speckle	-1.22534	-2.77200	1743	-0.47369	-0.64682	1436
		Gaussian	-1.68448	-3.35932	1471	-0.79105	-1.97890	1014
		Shot	-1.94548	-3.95596	1456	-0.47435	-0.74474	1166
kodim02	Low	Speckle	-1.48842	-0.75737	2121	-1.36064	-0.90849	1324
		Gaussian	-1.87576	-1.20941	1319	-2.11769	-1.33488	869
		Shot	-2.70271	-1.59829	1192	-0.80612	-0.37096	882
	High	Speckle	-1.57953	-0.54656	1117	-0.68677	-0.33656	887
		Gaussian	-2.64475	-1.06396	882	-2.19071	-0.72003	601
		Shot	-1.62973	-0.65146	762	-0.90468	-0.28160	615
kodim03	Low	Speckle	-1.65447	-0.74681	2581	-0.30322	-0.07469	2298
		Gaussian	-1.02335	-0.42048	2082	-0.50200	-0.18108	1620
		Shot	-1.59828	-0.51143	1645	-1.15672	-0.22841	1229
	High	Speckle	-1.27940	-0.15715	1400	-1.20334	-0.24147	904

		Gaussian	-1.98709	-0.25335	1275	-0.88843	0.05729	828
		Shot	-0.88261	0.19538	1450	-1.47964	0.10834	847
kodim12	Low	Speckle	-1.46139	-0.00491	2141	-1.0663	-0.17939	2825
		Gaussian	-1.06872	-0.33032	2041	-0.80805	-0.10259	2431
		Shot	-1.63745	0.10023	1717	-1.52984	-0.45597	1807
	High	Speckle	-3.10362	0.74423	1034	-3.05227	0.12072	1017
		Gaussian	-1.70767	0.13823	1370	-1.08043	0.12643	1599
		Shot	-2.76019	0.16165	1168	-2.63025	-0.00409	1312
Baboon	Low	Speckle	-7.8426	-22.26031	434	-4.60521	-12.68824	531
		Gaussian	-7.40393	-21.52717	498	-3.34769	-11.37382	733
		Shot	-7.77718	-22.27885	462	-4.51421	-12.52342	535
	High	Speckle	-6.64162	-20.50244	593	-4.2121	-12.20148	532
		Gaussian	-7.45738	-21.61997	367	-4.28815	-12.02414	518
		Shot	-6.38305	-19.69263	573	-3.8235	-12.38953	670
Average			-2.18683	-2.80937	1507.685	-1.51030	-1.57398	1313.500
Extra Iteration for Stagnation					1000			500
Total Number of Iterations					2507.685			1813.500

Note: SP refer to stopping point.

Table A-3: Overall Comparison of Performance of Speckle Denoising Models on CSet9 Images.

Image	Noise Level	Denoising Models					
		DD*		SGLD*		DIP*	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Baboon	Low	21.05359	0.63941	19.41878	0.47936	19.74172	0.50368
	High	20.63256	0.59591	19.15094	0.46952	19.32553	0.49387
F16	Low	26.27490	0.90613	27.99586	0.91194	28.36041	0.91800
	High	26.41065	0.91123	23.52331	0.86579	23.66905	0.87978
House	Low	30.38619	0.94975	32.67716	0.95561	32.32693	0.95314
	High	30.60149	0.94920	28.28808	0.93525	28.21635	0.94030
Lena	Low	28.83876	0.89055	29.09199	0.90596	29.05964	0.89012
	High	29.08991	0.89216	25.77598	0.86812	25.93133	0.86554
Peppers	Low	26.50702	0.86063	28.35424	0.88884	28.58978	0.88623
	High	26.33956	0.85968	26.04932	0.85330	26.14788	0.85759
kodim01	Low	23.37928	0.71661	25.83101	0.81426	26.67326	0.84105
	High	24.02313	0.76360	24.22916	0.74150	24.63131	0.77417
kodim02	Low	27.74385	0.85870	29.21487	0.86647	30.39825	0.88475
	High	27.59092	0.85430	28.86322	0.85756	28.52538	0.85450
kodim03	Low	28.59014	0.89542	29.56377	0.90259	30.65272	0.91574
	High	28.50474	0.88774	28.47510	0.88897	28.55639	0.89005
kodim12	Low	26.32414	0.87458	29.13812	0.89036	29.33887	0.88959
	High	26.40252	0.87314	25.99917	0.86277	26.37522	0.86504

Note: The superscript “*” indicates the MB-ES was applied to the model.

Table A-4: Overall Comparison of Performance of Super-Resolution Models on CSet9 Images. The best and second-best scores are highlighted in red and blue, respectively.

Image	Factor	Super-Resolution Models							
		Bicubic		SRCNN		ESPCN		DIP*	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Baboon	2	20.7695	0.7867	22.9102	0.8244	24.3335	0.8608	22.9956	0.7871
	3	20.9033	0.7088	21.3265	0.7172	22.1300	0.7520	20.4400	0.6484
	4	20.6022	0.6290	20.4878	0.6424	21.2311	0.6779	20.7211	0.6233
F16	2	24.8609	0.9262	29.3636	0.9570	32.9751	0.9790	31.4889	0.9666
	3	25.3046	0.9195	26.4625	0.9166	29.1376	0.9505	25.5693	0.9123
	4	25.0130	0.8995	24.5397	0.8650	27.2378	0.9205	27.0501	0.9238
House	2	30.9390	0.9659	34.5140	0.9790	42.5858	0.9967	40.3861	0.9870
	3	31.9799	0.9655	31.8093	0.9579	39.3516	0.9877	31.8456	0.9584
	4	31.6426	0.9557	29.5416	0.9198	35.3827	0.9725	35.6944	0.9678
Lena	2	27.4002	0.9198	31.2525	0.9499	34.9920	0.9692	32.8765	0.9470
	3	28.1882	0.9102	28.6310	0.9113	31.7397	0.9437	28.0248	0.9030
	4	28.0816	0.8929	26.8520	0.8639	29.8713	0.9187	29.8725	0.9097
Peppers	2	26.3416	0.9059	29.6982	0.9418	33.0547	0.9660	31.5948	0.9308
	3	26.4937	0.8971	27.3501	0.9054	30.5546	0.9462	26.1031	0.8878
	4	26.1771	0.8828	26.1205	0.8617	28.9430	0.9258	27.8420	0.9023
kodim01	2	22.3921	0.8015	24.4631	0.8330	26.8168	0.8801	25.8549	0.8546
	3	22.5176	0.7347	22.9444	0.7272	23.9494	0.7691	22.2905	0.7010
	4	22.1406	0.6642	21.9625	0.6426	22.7064	0.6854	22.7928	0.6911
kodim02	2	28.8643	0.9166	31.9016	0.9437	33.1831	0.9541	32.2053	0.9381
	3	29.2645	0.9017	30.1883	0.9105	31.0911	0.9235	28.4309	0.8874
	4	28.7402	0.8768	29.4551	0.8895	30.1688	0.9026	29.5725	0.8887
kodim03	2	28.9941	0.9330	31.8467	0.9534	34.1140	0.9697	33.0805	0.9591
	3	29.3750	0.9193	30.1685	0.9236	31.5655	0.9411	29.3823	0.9168
	4	28.9438	0.9019	28.9961	0.8916	30.4288	0.9205	30.2543	0.9164
kodim12	2	27.5453	0.9198	31.1125	0.9457	33.2345	0.9594	32.8173	0.9473
	3	27.9519	0.9031	28.4163	0.9050	30.2455	0.9264	27.6597	0.8978
	4	27.1571	0.8809	27.0239	0.8697	28.7153	0.8994	28.7106	0.8967

Note: The superscript “*” indicates the MB-ES was applied to the model.

Appendix B: Additional Figures

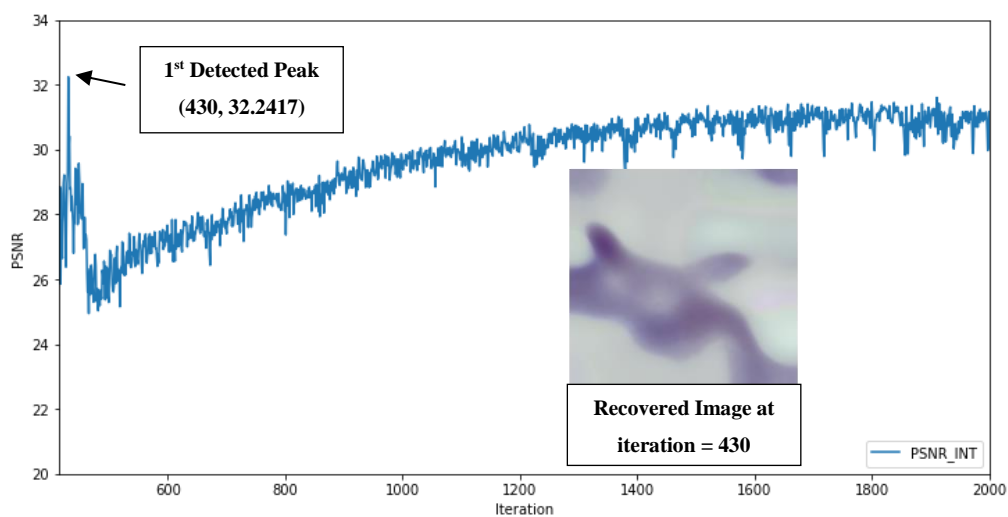


Figure B-1: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 1)

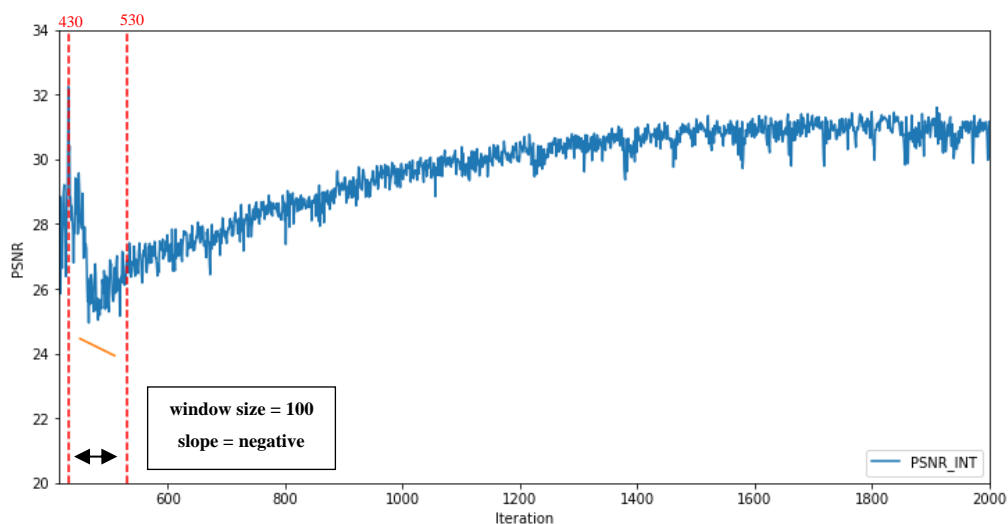


Figure B-2: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 2). Note: The window size multiplier in this algorithm is fixed at 100. The slope is computed from the best fit line, which is represented in orange.

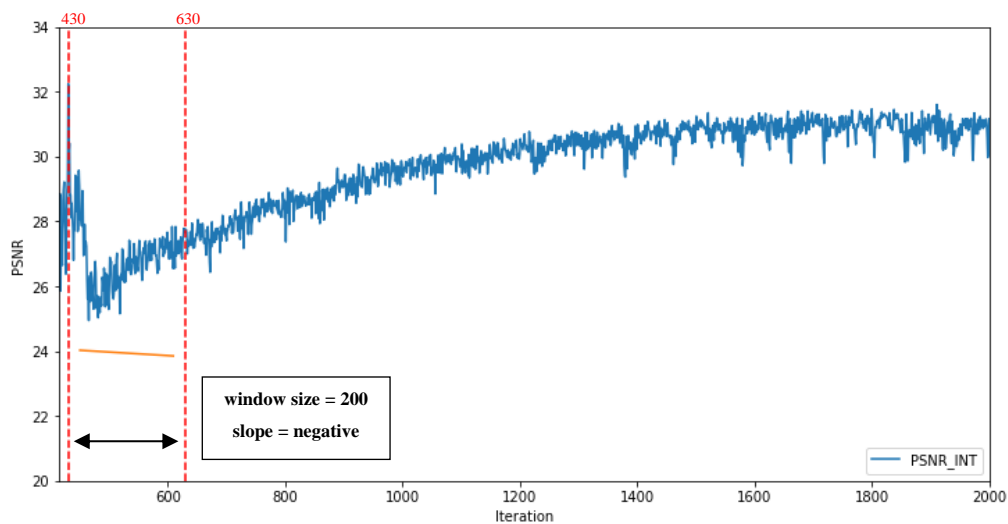


Figure B-3: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 3).

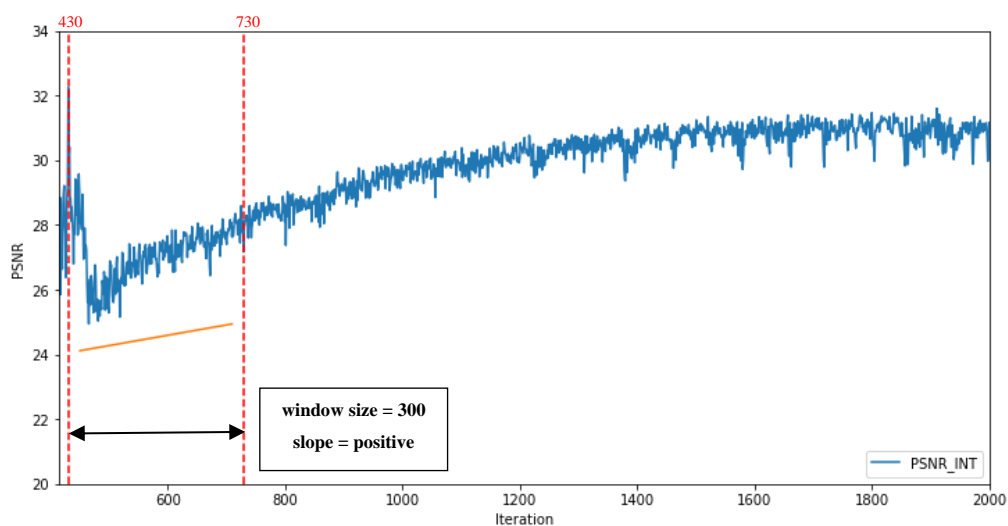


Figure B-4: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 4). Note: In cases where the slope is positive, the initial peak detected at iteration = 430 will be eliminated, and the algorithm will proceed to search for another peak.

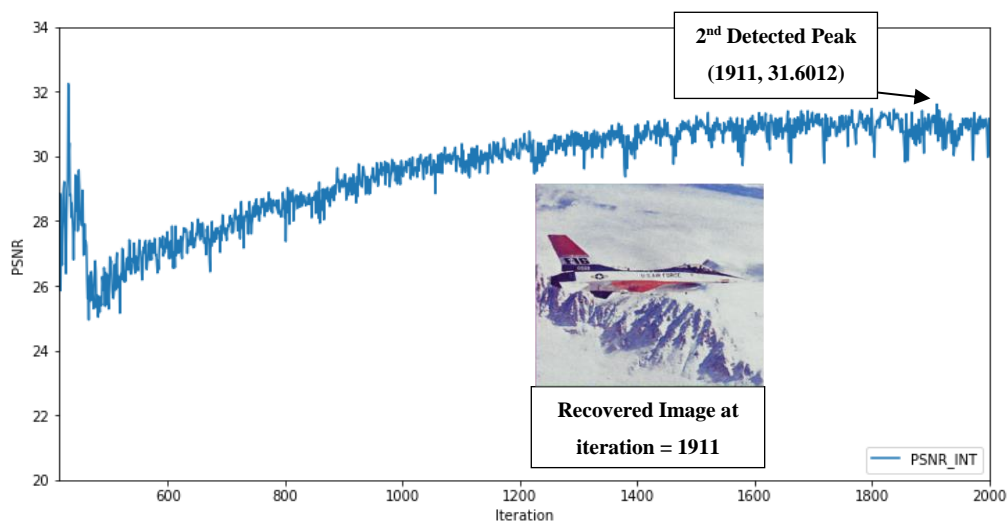


Figure B-5: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 5).

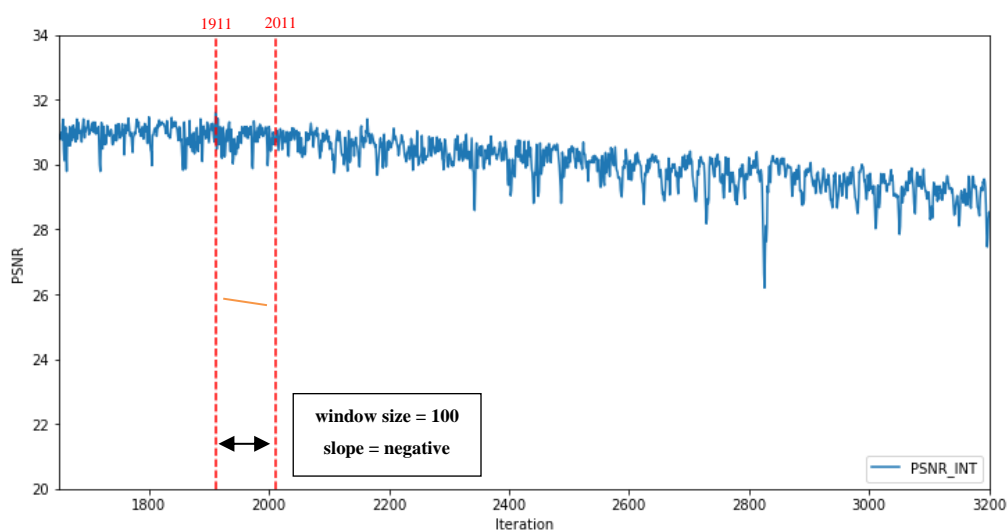


Figure B-6: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 6).

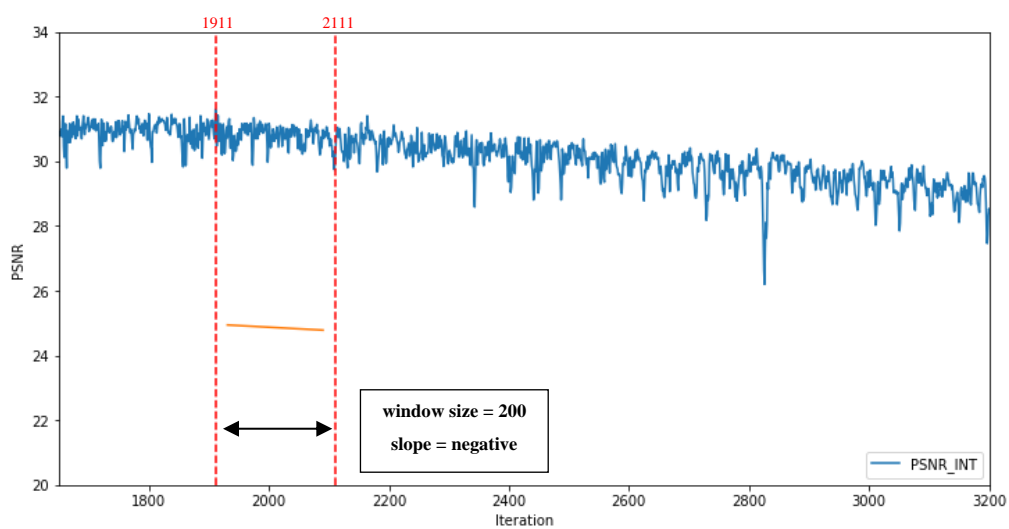


Figure B-7: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 7).

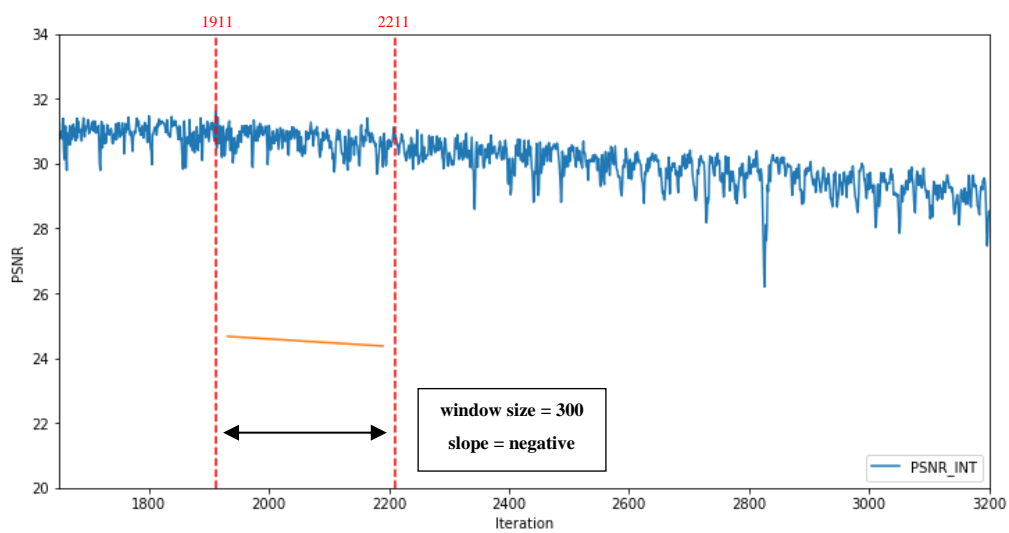


Figure B-8: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 8).

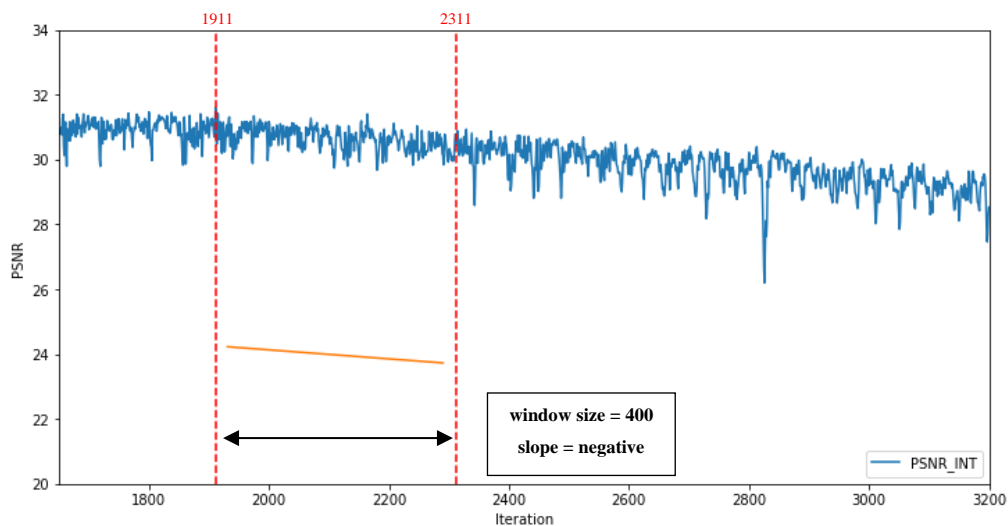


Figure B-9: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 9).

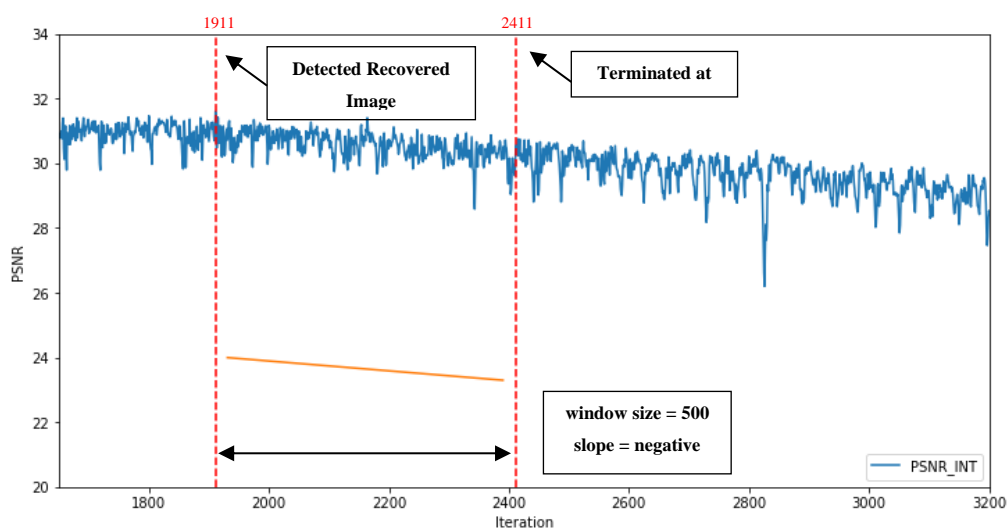


Figure B-10: Demonstration of Early Stopping Detection by Denoising-MB-ES (Stage 10). Note: The patience number is set to 5. As previously stated, the window size increases by 100 at each step. If the slope is negative for all windows (e.g., window sizes ranging from 100 to 500), the program will terminate at iteration = 2411 and return the image with the highest quality, which is obtained at iteration = 1911.

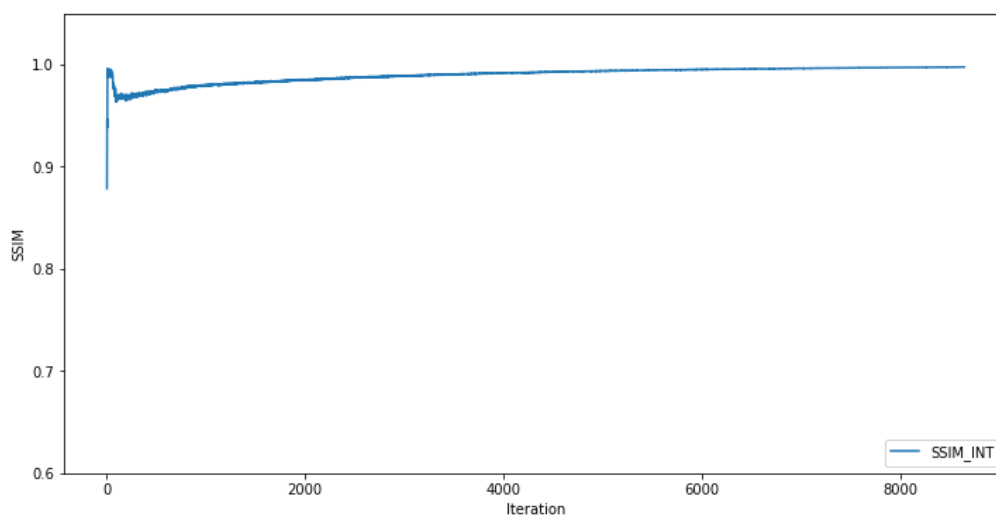


Figure B-11: Demonstration of Early Stopping Detection by Super-Resolution-MB-ES (Stage 1). Note: The window will calculate the slope for every iteration, but for the sake of simplicity, analysis will be performed selectively.

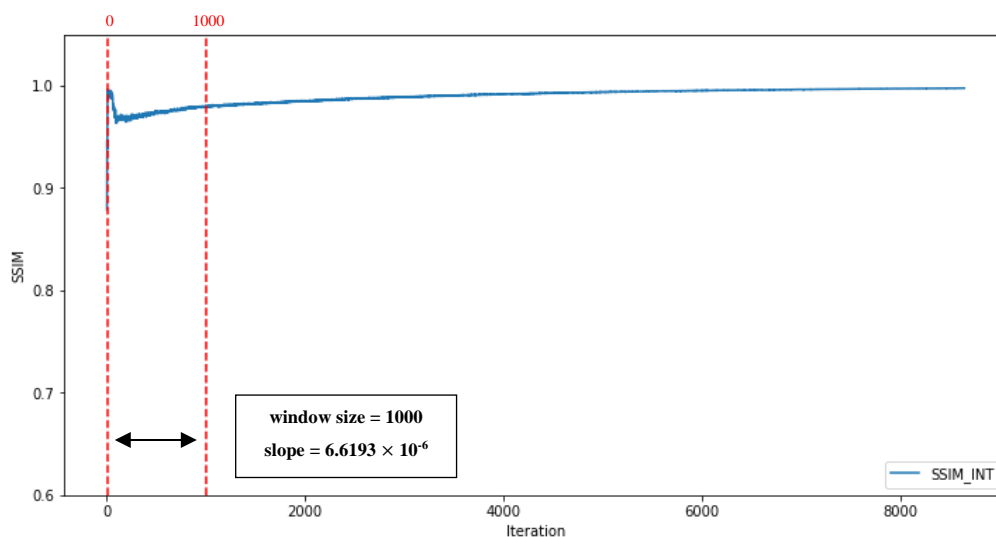


Figure B-12: Demonstration of Early Stopping Detection by Super-Resolution-MB-ES (Stage 2). Note: The computed slope is more than threshold of 2×10^{-6} .

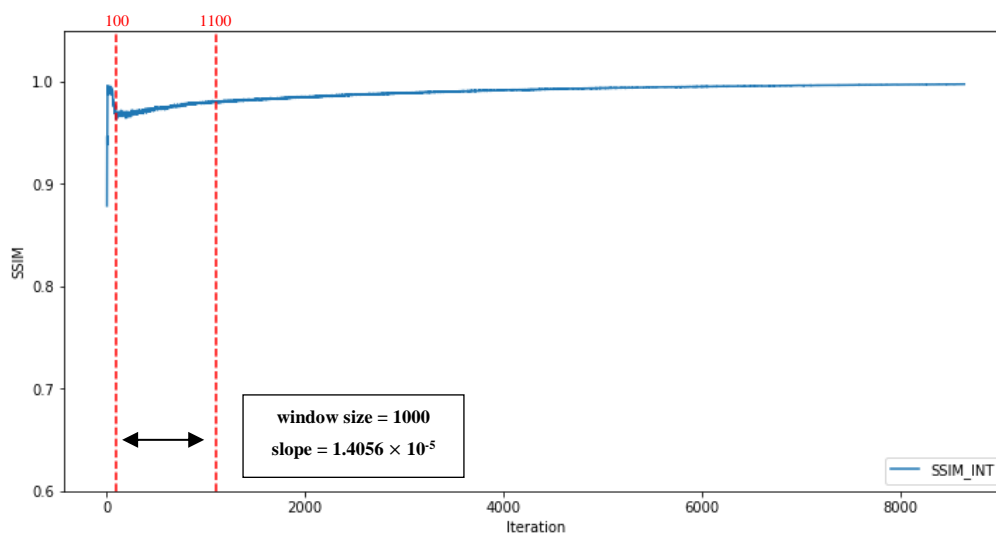


Figure B-13: Demonstration of Early Stopping Detection by Super-Resolution-MB-ES (Stage 3). Note: The computed slope is more than threshold of 2×10^{-6} .

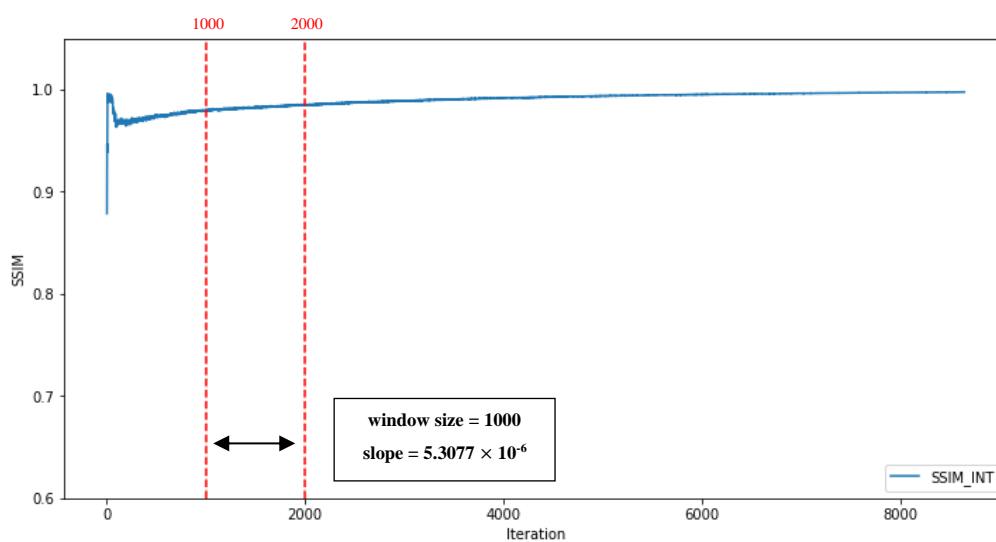


Figure B-14: Demonstration of Early Stopping Detection by Super-Resolution-MB-ES (Stage 4). Note: The computed slope is more than threshold of 2×10^{-6} .

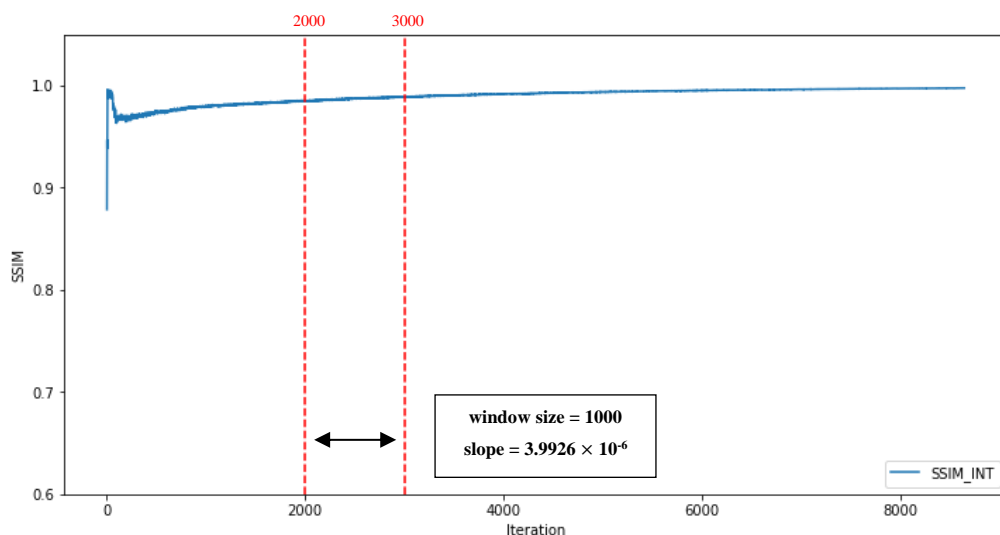


Figure B-15: Demonstration of Early Stopping Detection by Super-Resolution-MB-ES (Stage 5). Note: The computed slope is more than threshold of 2×10^{-6} .

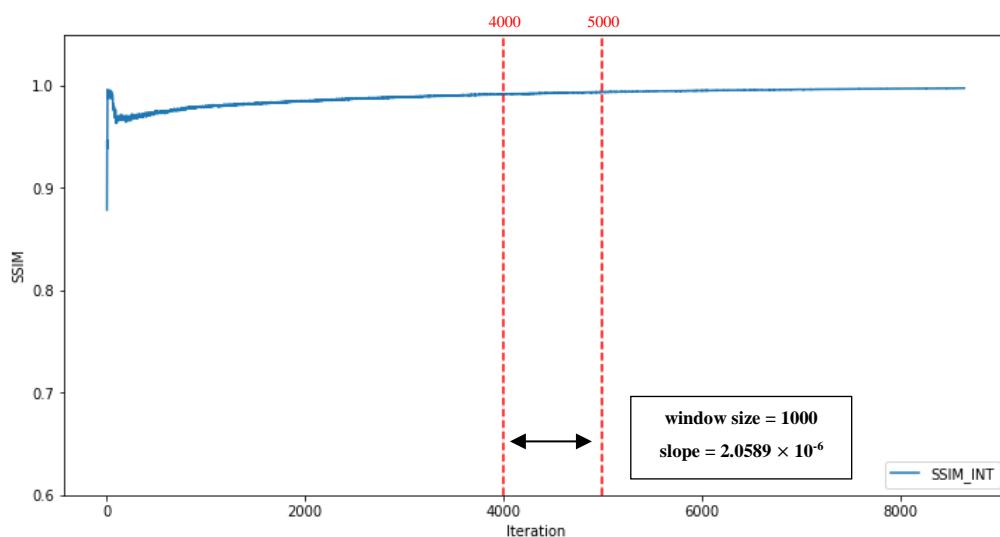


Figure B-16: Demonstration of Early Stopping Detection by Super-Resolution-MB-ES (Stage 6). Note: The computed slope is now approaching the threshold of 2×10^{-6} .

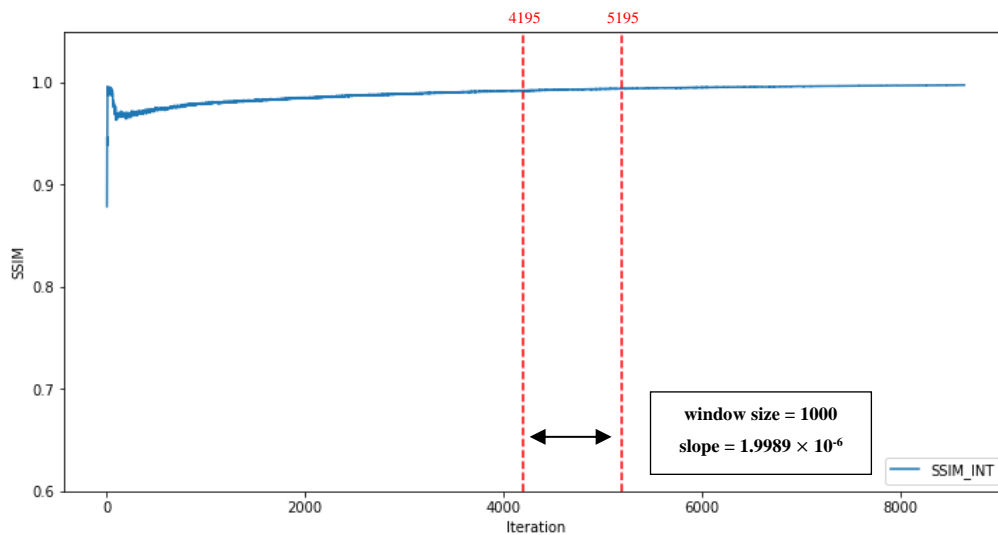


Figure B-17: Demonstration of Early Stopping Detection by Super-Resolution-MB-ES (Stage 7). Note: The computed slope is less than threshold of 2×10^{-6} . The “wait_count” will begin to increase.

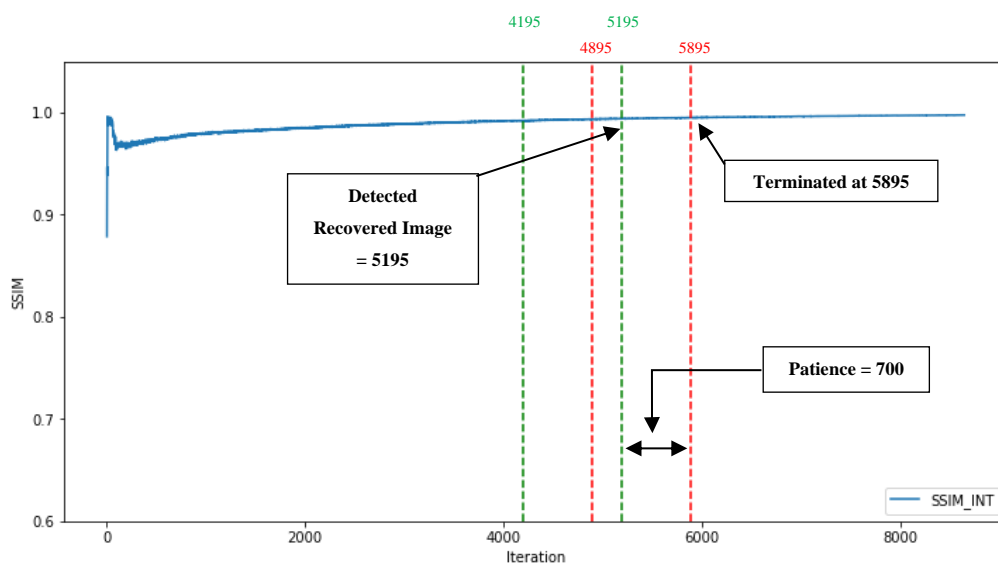


Figure B-18: Demonstration of Early Stopping Detection by Super-Resolution-MB-ES (Stage 8). Note: The “wait_count” equals to the “patience” of 700 at iteration = 5895.