ATTENTIVE MULTI-BRANCH ENCODER-DECODER NETWORK FOR ADHERENT OBSTRUCTION REMOVAL

ATTENTIVE MULTI-BRANCH ENCODER-DECODER NETWORK FOR ADHERENT OBSTRUCTION REMOVAL

By YUANMING CAO, B.Eng

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Applied Science

McMaster University

MASTER OF APPLIED SCIENCE (2023)

Hamilton, Ontario, Canada (Electrical and Computer Engineering)

TITLE:	Attentive Multi-Branch Encoder-Decoder Network for Adherent Obstruction Removal
AUTHOR:	Yuanming Cao B.Eng (Mechatronics Engineering), McMaster University, Hamilton, Canada
SUPERVISOR:	Dr.Xiaolin Wu
NUMBER OF PAGES:	xii, 73

Abstract

With the rapid development of image hardware, outdoor computer vision systems, for instance, surveillance cameras, have been extensively utilized for various applications. These systems typically equip a protective glass layer installed in front of the camera. However, during inclement weather conditions, images captured through such glass often suffer from obstructions adhering to its surface, such as raindrops or dust particles. Consequently, this leads to a degradation in image quality, which significantly affects the performance of the system.

Existing obstruction removal algorithms attempt to resolve these issues using deep learning techniques with synthetic data, which may not achieve a good visual result for complex real-world situations. To solve this, some studies employ real-world data. However, they tend to focus on a singular type of obstruction, such as raindrops.

This thesis addresses the more challenging task of restoring images taken through glass surfaces, which are impacted by various adherent obstructions such as dirt, raindrops, muddy raindrops, and other small foreign particles commonly found in real-life scenarios, including stone fragments and leaf particles. This work introduces an encoder-decoder network that incorporates auxiliary learning and an attention mechanism. During the testing phase, the auxiliary branch updates the shared internal hyperparameters of the model, enabling it to restore images from not limited to known categories of obstructions from

the training dataset, but also unseen ones. To better accommodate real-world situations, this work presents a dataset comprising real-world adherent obstruction pairs, which covers a large variety of common obstructions along with their corresponding clean ground truth images. Experimental results indicate that the proposed technique outperforms many existing methods in both quantitative and qualitative assessments.

"It's puzzling. I don't think I've ever seen anything quite like this before"

-Hal 9000, 2001: A Space Odyssey

Acknowledgements

First and foremost, I would like to express my profound gratitude and appreciation to my esteemed supervisor, Dr.Xiaolin Wu for his unwavering guidance, encouragement, and support throughout the journey of completing this thesis.

Furthermore, I would like to extend my heartfelt gratitude to my friends and family, particularly my parents, for their constant love and emotional support throughout the duration of my research studies.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
GAN	Generative Adversarial Network
MAE	Mean Absolute Error
MSE	Mean Square Error
MS-SSIM	Structural Similarity Index Measure
OpenCV	Open Source Computer Vision Library
PSNR	Peak Signal-to-Noise Ratio
ResNet	Residual Network
SSIM	Multi-Scale Structural Similarity

Declaration of Academic Achievement

A portion of the research presented in Chapters 5, 6, and 7 has been submitted to the International Conference on Computer Vision 2023:

Qiang Li, Yuanming Cao, Zhixiang Chi and Xiaolin Wu, "Seeing through Unclear Glass: Occlusion Removal with One Shot"

As the co-first author of the paper, I was responsible for the majority of data acquisition, experimentation, and network implementation. Zhixiang Chi and I collaborated on the network design. Qiang Li contributes to the development of the adherent obstruction mathematical model and the majority of the writing manuscript. Dr. Xiaolin Wu serves as the supervisor for this project, offering invaluable guidance and assistance.

Chapter 1

Introduction

The rapid advancement of image hardware technology has led to the increasing implementation of outdoor vision systems across various applications. These systems are commonly equipped with a protective glass layer to prevent them from external elements and potential damage. Over time, or under extreme weather conditions, this glass layer may accumulate various types of occlusions, such as mud, and small particles. Consequently, images captured through such an obscured medium are subject to multiple forms of degradation, such as hazing, and occlusion. These degraded images significantly affect the reliability of outdoor vision systems, potentially resulting in false positives or false negatives, which in turn may lead to inaccurate object detection outcomes in applications such as autonomous vehicle systems. As a result, there is a need to develop algorithms that recover images free from obstructions.

Addressing the challenge of restoring images affected by adherent obstruction to their original, clean state is a problem within the domain of image restoration. As a low-level vision task, image restoration involves the process of recovering images that are free from degradation factors, such as addictive noise and artifacts. These techniques frequently

serve as pre-processing methods that can enhance the performance of high-level vision tasks, ultimately improving the effectiveness of computer vision systems [16].

In the early stages of image restoration research, researchers employed spatial domain and frequency domain methods that relied on simple mathematical models to reduce noise and artifacts in degraded images, such as median filtering and Wiener filtering [16]. As the field advanced, more sophisticated methods were introduced, including model-based methods like Total Variation Regularization [49] and Non-Local Means [5]. In addition to using model-based approaches, some researchers also employed external image priors and sparse representations [12]. Although these approaches demonstrate satisfactory performance in certain situations, they are often time-consuming, computationally demanding, and inadequate in effectively restoring complex image textures.

The advent of deep learning and artificial neural networks revolutionizes the field of image restoration. With the rapid evolution of artificial neural networks, the scope of degradation types of image restoration problems expands from addictive noise to more diverse and complex ones such as deraining, obstruction removal, and image inpainting [61, 9, 37]. As a result, the capabilities of image restoration techniques have significantly improved, offering more robust solutions for a wider range of real-world challenges.

This thesis mainly focuses on the image restoration problem which removes artifacts caused by images taken through obscured glass. To address this prevalent issue, this thesis presents an attention-aided multi-branch network for adherent obstruction removal problem. By integrating auxiliary learning into the network, it gains the ability to adapt to previously unencountered types of obstructions. Furthermore, this work presents a real-world dataset for the adherent obstruction removal problem to enhance the network's performance. Consequently, the proposed method successfully and efficiently reconstructs

obstructed regions while maintaining the visual context and integrity of the scene.

1.1 Contribution

This thesis presents a learning-based approach for adherent obstruction removal, along with auxiliary learning and attention mechanism. Inspired by the work done by Chi et al[7], this work implements a two-branch network that enables test-time adaptation. Apart from that, this work also proposes a real-world dataset for the adherent occlusion removal problem. The dataset includes several common obstructions such as dirt, raindrops, muddy raindrops, and small particles. This thesis makes the following contributions:

- This thesis discusses a new challenge for the image restoration problem, namely, the adherent obstruction removal problem.
- This study introduces a real-world dataset tailored for addressing adherent obstruction removal challenges.
- This work proposes and implements a learning-based approach for tackling obstruction removal challenges, namely attentive multi-branch encoder-decoder network for adherent obstruction removal.

1.2 Programming Language and Libraries

1.2.1 Python

Due to its straightforward and intuitive syntax, this work chooses Python as the programming language for the development of all codes.

1.2.2 PyTorch

PyTorch[1] is an Python open-source machine-learning library which is extensively employed in the development of deep learning applications. PyTorch provides a flexible and dynamic approach to building and training neural networks. It is built around a central data structure called a tensor. The data structure of the tensor is closely resembles NumPy's ndarray (Matrix Form) but is specifically optimized for GPU acceleration(especially for NVIDIA GPU). PyTorch provides a range of functions for creating and manipulating tensors, including functions for linear algebra operation and Fourier transforms[58].

1.2.3 NumPy

NumPy[2] is a Python library dedicated to numerical computing, expecially for matrix computation. Numpy offers a robust array data structure accompanied by an extensive array of functions for data manipulation. NumPy also provides a diverse set of functions for array operations, such as array creation, reshaping, slicing, and indexing.

1.2.4 OpenCV

OpenCV (Open Source Computer Vision Library)[3] is an Python open-source computer vision library for both Python nad C/C++ programming languages. It encompasses a comprehensive selection of image processing algorithms and functions, including decoding images into matrix forms, performing edge detection, and changing the color space of image matrices.

[1] https://pytorch.org
[2] https://numpy.org
[3] https://opencv.org

Chapter 2

Background

2.1 Machine Learning Background

This section delivers a comprehensive overview of essential concepts and background information for the development of the proposed learning-based method.

2.1.1 Machine Learning

Machine learning constitutes a subdomain of artificial intelligence that focuses on developing algorithms and models that empower computers to learn, as well as to make predictions based on provided data. It builds on the idea that machines can automatically learn to recognize complex patterns and make intelligent decisions without being explicitly programmed to do so. This is achieved through constructing mathematical models that are trained on sample data known as training data, allowing them to adapt their behavior based on the input they receive[30, 40].

2.1.2 Deep Learning

Although deep learning and machine learning are often used interchangeably, it is essential to acknowledge the distinctions between these two concepts. Machine learning, deep learning, and neural networks each represent subdomains within the artificial intelligence field. Neural networks are actually a sub-field of machine learning, and deep learning is a sub-field of neural networks. Deep learning also known as deep neural networks, consists of multi-layers (usually more than three layers). The key advantage of deep learning over traditional machine learning is its ability to automatically learn features from raw input data, eliminating the need for extensive manual feature engineering. This capability is particularly useful when working with large amounts of high-dimensional data, such as images, text, or speech signals[17].

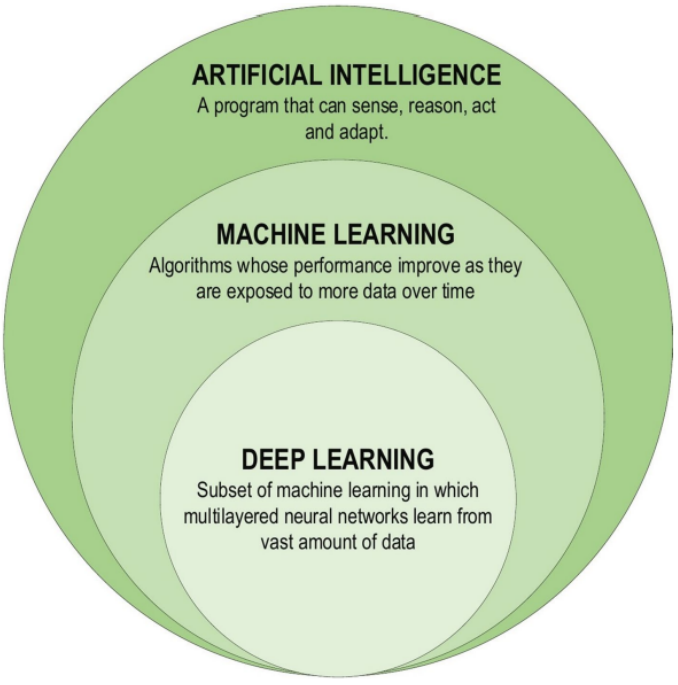


Figure 2.1: An illustration for deep learning family[33]

2.1.3 Supervised/Unsupervised Learning

There are several types of machine learning approaches, such as supervised learning and unsupervised learning. Supervised learning entails acquiring knowledge from labeled data, wherein the algorithm receives input-output pairs and aims to learn a mapping between them. Given a collection of input-output pairs with size N, denoted as $(x_1, y_1), ..., (x_N, y_N)$, the i-th input data, represented by x_i is referred to as the feature vector, while y_i signifies the corresponding i-th label. The supervised learning model endeavors to identify a mapping function, or hypothesis function, expressed as $g : X \rightarrow Y$, where X symbolizes the input space and Y represents the output space[40]. The learning process entails the discovery of an appropriate hypothesis function g within a predetermined space of potential functions, designated as $\mathbf{G}$, which is commonly referred to as the hypothesis space. The hypothesis function g can also be represented using a scoring function $f : X \times Y \rightarrow \mathbb{R}$, where g represents the returned y value that corresponds to the highest score achieved:

$$g(x) = \underset{y}{\operatorname{argmax}} f(x, y) \tag{2.1.1}$$

In contrast to supervised learning, the model for unsupervised learning earns to identify patterns or structures by analyzing the relationships between data points, without using any predefined target variable. The primary goal of unsupervised learning is to discover hidden structures, patterns, or relationships within the data, without relying on any prior information or guidance[3]. Self-supervised learning is a subcategory of unsupervised learning in which a model learns to make predictions or representations from the input data without relying on human-labeled examples. Instead, the learning process leverages the structure and inherent properties of the data itself to generate ”pseudo-labels” that the model can use

for training [26].

2.1.4 Auxiliary Learning and Test-time Adaptation

Auxiliary learning, alternatively referred to as multi-task learning, is a training methodology wherein a singular model is concurrently trained to execute multiple correlated tasks[38]. The fundamental concept for auxiliary learning is the exploitation of shared representations or features across tasks, thereby enhancing the model's generalization and increasing the performance of the primary task. By acquiring knowledge from auxiliary tasks, the model can adeptly capture and leverage the underlying data structure, which may remain obscured when concentrating solely on the primary task.

In the context of auxiliary learning, the model is structured with multiple output branches, primary branches and auxiliary branches. The primary branch is the main objective of the model, while auxiliary branches aim to assist the primary branch. By carefully choosing correlated tasks for the auxiliary branch, the trained model is able to adapt to unseen data without any additional training or fine-tuning of the primary task, this technique is known as test-time adaptation. By leveraging auxiliary learning, the model can be guided to adjust the shared internal parameters of both the auxiliary branch and primary branch, resulting increase in performance on the primary task at test time[7]. This is particularly useful when the test data distribution differs from the training data distribution[13].

2.1.5 Optimization

In machine learning, the objective is to develop a model capable of generating accurate predictions given input data. To achieve this, a loss function, denoted as $J(\theta)$ is established to quantify the difference between the predicted outcomes and the true labeled data.

θ represents the model parameters: $\theta \in \mathbb{R}^d$. Given the learning rate η, which determines the size of steps to reach a minimum, The goal is to find the model parameters θ^* that minimize the loss function, and this is the optimization process. Optimization in machine learning can be broadly classified into two categories: convex and non-convex optimization. Convex optimization pertains to scenarios when every local minimum concurrently serves as a global minimum, and the objective function is convex. Conversely, non-convex optimization addresses issues where the objective function is non-convex and may possess multiple local minima[4]. Machine learning models, especially deep learning models, often involve non-convex optimization due to the complexity of their loss functions. To solve these optimization problems, various optimization algorithms have been proposed[48]:

Batch gradient descent: Batch gradient descent, also known as vanilla gradient descent, is a first-order optimization algorithm that relies on the gradient of the loss function to update the model parameters. It calculates the gradient for the entire dataset and updates the model parameters accordingly:

$$\theta = \theta - \eta \cdot \boldsymbol{\nabla}_\theta J(\theta) \tag{2.1.2}$$

The main drawback of gradient descent is its computational cost when dealing with large datasets.

Stochastic Gradient Descent: Stochastic Gradient Descent (SGD) is a variant of gradient descent that updates the model parameters using the gradient of the loss function with respect to a single data point, chosen randomly at each iteration. This makes the algorithm much faster than gradient descent but introduces more noise in the optimization process.

$$\theta = \theta - \eta \cdot \boldsymbol{\nabla}_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.1.3}$$

Mini-batch Gradient Descent: Mini-batch gradient descent is a compromise between gradient descent and SGD. It updates the model parameters using the gradient of the loss function with respect to a mini-batch of data points, which reduces the noise of SGD while maintaining faster convergence compared to full gradient descent.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.1.4}$$

Adaptive Learning Rate Method: Adaptive Learning Rate Methods, such as Adaptive Moment Estimation (Adam)[29], Adagrad[10], and RMSprop[1], adjusting the learning rate during optimization. They maintain per-parameter learning rates, allowing for individual updates of each model parameter depending on its importance. This helps improve convergence speed and performance in complex optimization landscapes. Optimization in machine learning often involves additional techniques to improve generalization and prevent over-fittings, such as regularization (e.g., L1, L2, or dropout[54]) and early stopping.

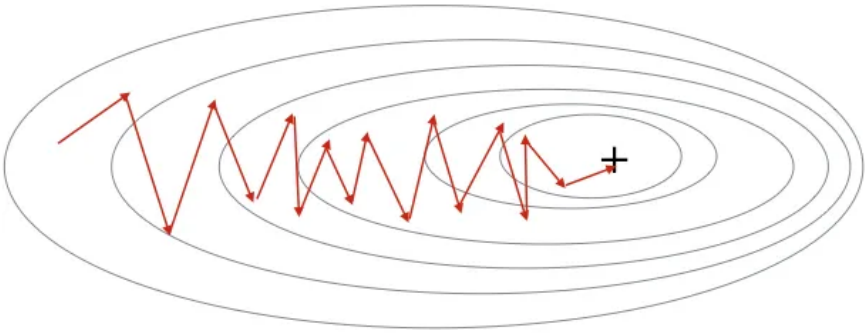


Figure 2.2: A graphical representation of a optimization process using Stochastic Gradient Descent[25]

[1]RMSprop proposed by Geoff Hinton is an unpublished adaptive learning rate technique. This technique is introduced during his Coursera class in Lecture 6e.

2.1.6 Backpropagation

In the gradient descent process, the gradient of the loss function,$\nabla_\theta J(\theta)$, is required. Although determining this gradient using analytical expressions is relatively simple, each gradient descent step requires the calculation of gradients for all weights and biases (parameters) within the network. Consequently, for extensive networks, such as those comprising one million parameters, the associated computational expense is considerably high. In 1986, Rumelhart et al. [50] conducted an experimental analysis of backpropagation applied to several neural networks. Their findings demonstrated that backpropagation significantly outperformed the direct computation approach, making previously intractable problems solvable.

The operations of the Backpropagation neural networks can be divided into two steps: feedforward and Backpropagation[42]. In the feedforward step, the input is passed through the network to compute the predicted output. Each neuron calculates a weighted sum of its inputs, adds a bias term, and applies an activation function. The predicted output O, also known as the activation, is then compared to the true output, and the loss is computed using a loss function. The backward pass calculates the gradients of the loss with respect to the weights and biases of the network. Using the chain rule, these gradients are computed in a layer-wise manner, starting from the output layer and moving back toward the input layer. The gradient calculation involves computing the partial derivatives of the loss function with respect to the neuron's activation and the weighted sum at each layer. Given l as the last layer of the neural network, a cost function C, a bias b^l, and a weight w^l with respect to l, the gradient can be calculated as:

$$\frac{\partial C}{\partial w^l} = \frac{\partial C}{\partial a^l}\frac{\partial a^l}{\partial z^l}\frac{\partial z^l}{\partial w^l} \tag{2.1.5}$$

$$\frac{\partial C}{\partial b^l} = \frac{\partial C}{\partial a^l}\frac{\partial a^l}{\partial z^l}\frac{\partial z^l}{\partial a^l} \tag{2.1.6}$$

where $z^l = w^l * x + b$ and $a^l = f(z^l)$ with f bing the activation function. The weight and bias for the last layer of nerual net can be updated using an optimization technique discussed in the previous section.

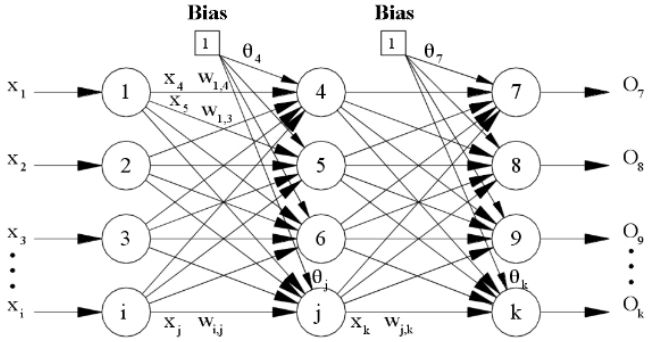


Figure 2.3: Example of Backpropagation Neural Network with one hidden layer[39, 42]

2.1.7 Neural Network

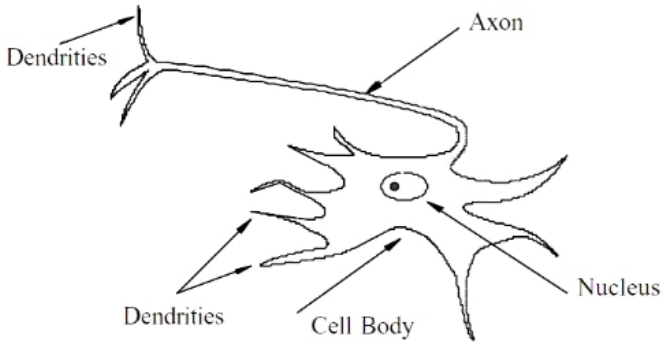


Figure 2.4: Anatomy of human neuron parts[39, 41]

The Neural network in machine learning, often referred to as an artificial neural network (ANN), is a computational model inspired by the biological neural networks found in the human brain[14]. Similar to a biological human neuron shown in Figure.2.4, the artificial neural network is a simplified abstraction. In the human brain, neurons are the basic building blocks that receive, process, and transmit information through electrical and chemical signals[39]. Similarly, ANNs consist of interconnected processing units called artificial neurons or nodes. Each artificial neuron receives input from other neurons, performs a computation, and sends the output to subsequent neurons in the network. The connections between artificial neurons are like dendrites and axons[41], with each connection having an associated weight that models the strength of the synapse. These weights determine the influence of one neuron's output on another neuron's input. When the integrated signal in a biological neuron reaches a certain threshold, the neuron generates an action potential or spike, which is then transmitted along the axon to other neurons. In artificial neurons, this behavior is modeled by applying a non-linear activation function to the weighted sum,

determining the neuron's output. Common activation functions include the sigmoid, hyperbolic tangent, and rectified linear unit (ReLU), which introduce non-linearity into the network, allowing it to learn complex patterns.

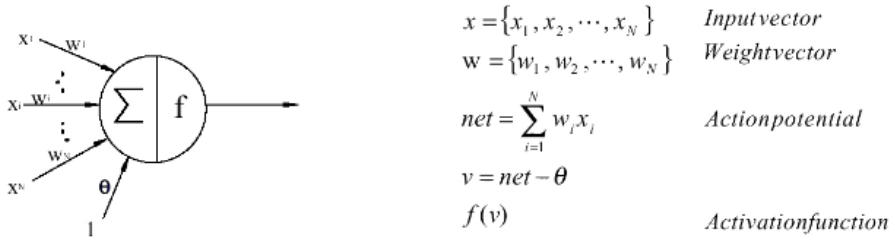


Figure 2.5: An example of an artificial neuron, this neuron calculates the weighted sum of its inputs, followed by a non-linear function.[39, 41]

2.1.8 Convolutional Neural Network

Convolutional Neural Network (CNN) proposed by Lecun et al. [34] is a Deep Learning algorithm that is specifically designed for tasks that involve processing grid-like data structures such as images. CNNs leverage the spatial structure of the input data by employing convolutional layers. In each layer, a filter, often referred to as a kernel, which is a $n \times m \times d$ matrix (usually a square matrix), is placed over a small region of the input data, originating from the top-left corner of the input matrix. The element-wise product between the filter and the input region it covers is computed, followed by summing up the resulting values. This sum produces a single value, which becomes the output for that specific region. Then, the filter slides across the input data, typically by one or more pixels at a time (known as the stride). At each position, the convolution process is repeated to compute the corresponding output value, until it reaches the end of the input matrix. As the filter progresses across the entire input data, a new matrix denoted as the feature map (or convolutional layer output)

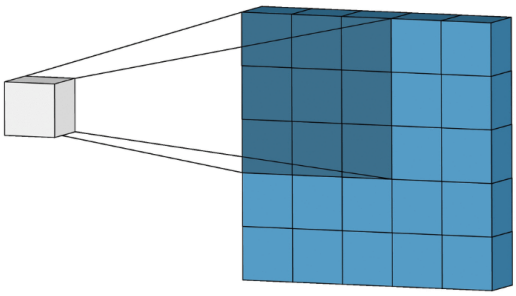


Figure 2.6: Illustration of convolution process using 3x3 kernel[52]

is created. Given the 2D input matrix I with size $m \times n$ and a filter F,the feature map g can be calculated as:

$$g(x,y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} I(x+i, y+j) \cdot F(i,j) \tag{2.1.7}$$

Assuming that both the input and filter are square matrices, let the dimension of the input matrix be M and the dimension of the filter be F.The size of the feature map can be calculated as:

$$M_{out} = \left\lfloor \frac{M - F + 2P}{S} \right\rfloor + 1 \tag{2.1.8}$$

Such feature map encapsulates the spatial information and discernible patterns inherent in the input data, thereby serving as a vital component in the analysis and interpretation of complex structures. By stacking multiple convolutional layers, CNNs can learn increasingly complex and abstract features. CNN architectures also incorporate activation layers, pooling layers, and fully connected layers for non-linearity, dimensionality reduction, and final output generation, respectively [31]. The network is trained using supervised learning algorithms and backpropagation to minimize a loss function, which measures the difference between its predictions and the true output labels.

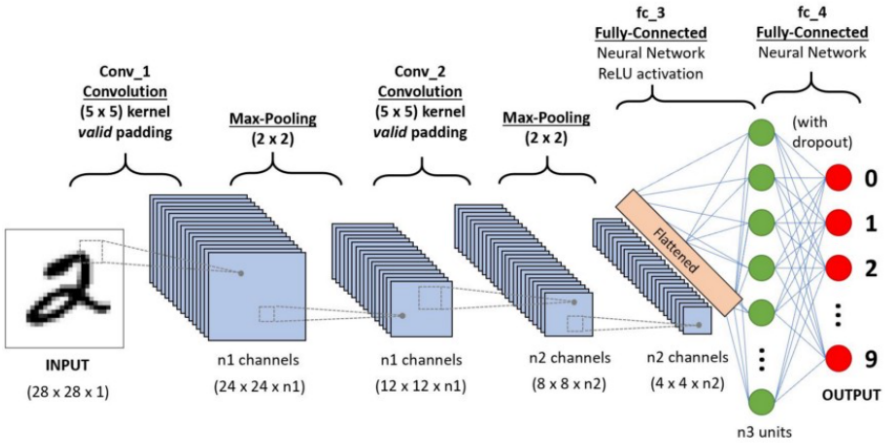


Figure 2.7: An example of CNN architecture, LeNet[34]

2.1.9 Residual Network

ResNet, an abbreviation for Residual Network, represents a deep neural network architecture proposed by He et al. in their paper, "Deep Residual Learning for Image Recognition" [23]. Training deeper neural networks can be challenging due to the problem of vanishing gradients. To address this issue, ResNet introduces shortcut connections, also known as residual connections, which allow deep neural networks to learn a residual function, effectively mitigating the vanishing gradient problem and facilitating more effective training. In a ResNet, each residual block comprises a sequence of convolutional layers, followed by batch normalization [2]and activation functions. The residual block can be expressed mathematically as:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{\mathcal{W}\}) + \mathbf{x} \tag{2.1.9}$$

[2]Batch Normalization [24] computes the mean and variance of each equally divided subset (mini-batch), then normalizes the a layer's input by subtracting the mean and dividing by the standard deviation. This technique mitigates internal covariate shift, and accelerates convergence.

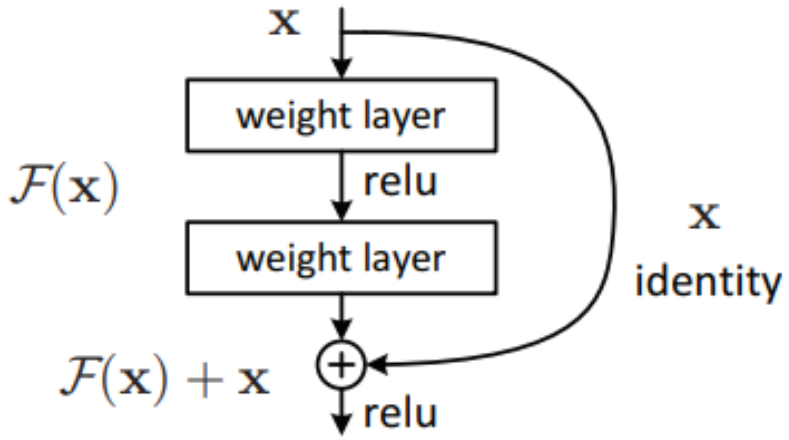


Figure 2.8: Residual Block with shortcut connection[23]

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output of the residual block, $\mathcal{F}(\mathbf{x}, \{\mathcal{W}\})$ represents residual mapping that needs to be learned

2.1.10 U-Net

U-Net[47] is a type of convolutional neural network (CNN) architecture that was originally designed for biomedical image segmentation tasks. It was introduced by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in their 2015 paper "U-Net: Convolutional Networks for Biomedical Image Segmentation.". Inspired by the shape of the English alphabet 'U', the U-Net architecture has a symmetric design. With left half of the 'U" as the encoder path and the right half as the decoder path.

The encoder path is composed of several layers of convolutional blocks, each consisting of two consecutive 3x3 convolution layers followed by a rectified linear unit (ReLU) activation function and a 2x2 max-pooling layer for downsampling. As the path progresses, the number of feature channels is doubled after each max-pooling layer, allowing the network to learn increasingly complex features and capture hierarchical information.

The decoder path mirrors the contracting path but replaces the max-pooling layers with 2x2 up-convolution (also known as transposed convolution) layers for upsampling. Each up-convolution layer is followed by a concatenation operation with the corresponding feature map from the contracting path, which is achieved via skip connections. This process helps the network retain finer-grained spatial information. After concatenation, two 3x3 convolution layers and ReLU activations are applied, similar to the contracting path.

The output from the deocder path's last layer is passed through a 1x1 convolution layer with a softmax or sigmoid activation function, generating the final segmentation map. This map represents the probability of each pixel belonging to a specific class or region in the input image.

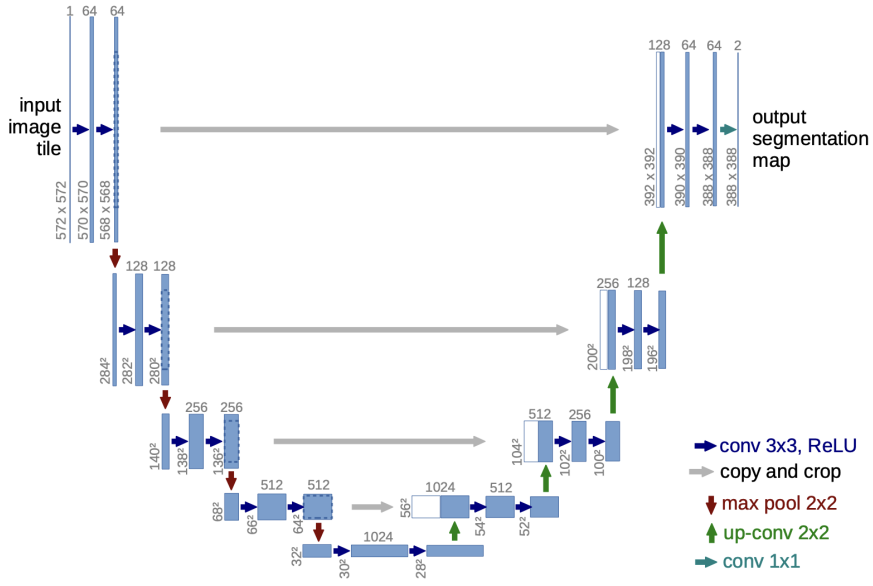


Figure 2.9: U-Net Architecture[47]

In 2015, the U-Net architecture demonstrated superior performance compared to the previously established best method, a sliding-window convolutional network, on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks [47]. In

addition to its achievements in the field of image segmentation, the U-Net architecture exhibits versatility by accommodating adjustments in the activation function, allowing for its application in various image restoration tasks. For instance, U-Net has been employed in image denoising, as demonstrated by Abascal et al. [1], and in image deblurring, as shown by Chi et al. [7]. These examples highlight the flexibility and adaptability of the U-Net architecture, enabling its use across a diverse range of image processing challenges.

2.1.11 Autoencoder

Autoencoders are a type of unsupervised neural network architecture that can learn efficient representations of input data by compressing and reconstructing the input through a bottleneck layer[63]. The presence of a bottleneck in the network force a compressed knowledge representation of the original input, which could potentially pose challenges during the reconstruction process.

In the case where input features are entirely independent of one another, achieving accurate compression and subsequent reconstruction becomes a complex task. However, if the data exhibits some inherent structure, such as correlations between input features, the network can effectively learn and exploit this structure while passing the input through the bottleneck[62].

Like U-Net, Autoencoder consists of an encoder and decoder network, the encoder is composed of several layers of feedforward neural networks or convolutional neural networks (CNNs). For image data, CNNs are often used to extract spatial information and hierarchical features. As the encoder progresses, the spatial dimensions are typically reduced using pooling or strided convolutions, while the number of feature channels increases. The final layer of the encoder produces the latent representation, often referred to as the ”code”

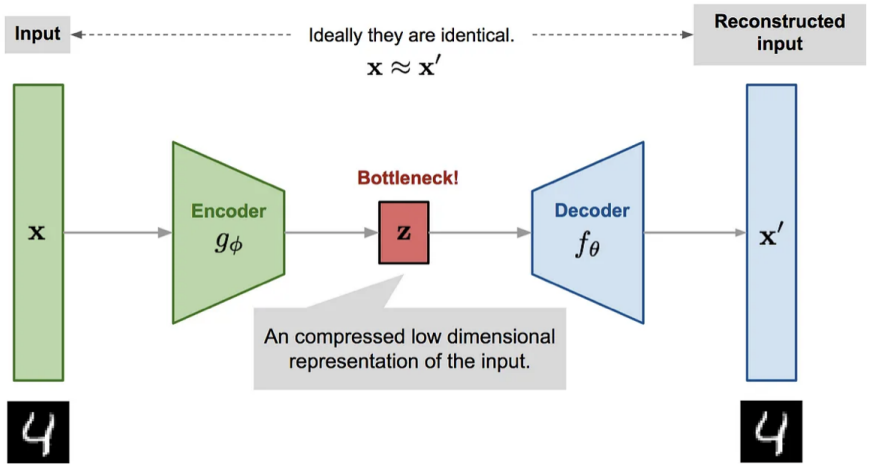


Figure 2.10: **An autoencoder architecture.** The input image undergoes a systematic encoding process, resulting in a compact representation and then be decoded. [62]

or "bottleneck." The decoder is designed to mirror the encoder's structure but in reverse order. It takes the latent(compressed) representation as input and attempts to reconstruct the original data. For image data, the decoder often employs transposed convolutional layers or upsampling operations to increase the spatial dimensions while reducing the number of feature channels. The final layer of the decoder outputs the reconstructed data, typically using an activation function that matches the desired range of the input date.

Mathematically, the proposed model comprises an encoder function, denoted by $g(\cdot)$ and parameterized by ϕ. A decoder function, denoted by $f(\cdot)$ and parameterized by θ. The compact representation, acquired from the input $\mathbf{x}$ within the bottleneck layer is represented as $\mathbf{z} = g_\phi(\mathbf{x})$. Consequently, the reconstructed input is expressed as $\mathbf{x}' = f_\theta(g_\phi(\mathbf{x}))$.

The parameters (θ, ϕ) are jointly optimized to produce a reconstructed data sample that closely approximates the original input, $\mathbf{x} \approx f_\theta(g_\phi(\mathbf{x}))$. In essence, this process aims to learn an identity function[62]. To quantify the discrepancy between two vectors, various

metrics can be employed, such as mean squared error (MSE) loss:

$$L_{\text{AE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}^{(i)} - f_\theta(g_\phi(\mathbf{x}^{(i)})))^2$$

2.1.12 Attention

During human cognitive processes, our brain can selectively concentrate on specific aspects of a scene while filtering out less relevant details. Inspire by this idea, the attention mechanism is created. the attention mechanism allows models to selectively focus on specific parts of an image or specific features when making predictions or generating output. For instance, the Convolutional Block Attention Module (CBAM)[65] focuses on refining feature representations by applying two types of attention mechanisms: channel attention and spatial attention. Channel attention focuses on emphasizing informative feature channels. It uses global average pooling and global max pooling to capture channel-wise statistics, followed by a shared multi-layer perceptron (MLP) to generate channel attention weights. These weights are then used to rescale the input feature maps. The spatial attention focuses on emphasizing relevant spatial locations in the feature maps. It uses global average pooling and global max pooling to capture spatial information from the input feature maps, followed by a convolutional layer with a kernel size of 7x7 to generate a spatial attention map. This map is then used to rescale the input feature maps element-wise.

Given an intermediate feature map $\mathbf{F}$ with channel size of $\mathbf{C}$, height of $\mathbf{H}$ and width of $\mathbf{W}$ as input, $F \in \mathbb{R}^{C \times H \times W}$, the attention process can be summarized as:

$$\mathbf{F}' = \mathbf{M_s}(\mathbf{F}) \otimes \mathbf{F} \tag{2.1.10}$$

Where $\otimes$ denotes element-wise multiplication and $\mathbf{M_s}$ is the 2D Spatial Convolutional

Block Attention Module:

$$\begin{aligned} \mathbf{M_s}(\mathbf{F}) &= \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times7}([\mathbf{F}^s_{avg}; \mathbf{F}^s_{max}])) \end{aligned} \tag{2.1.11}$$

whereσ denotes the sigmoid function and $f^{7\times7}$ is a convolution filter with size 7x7[65]

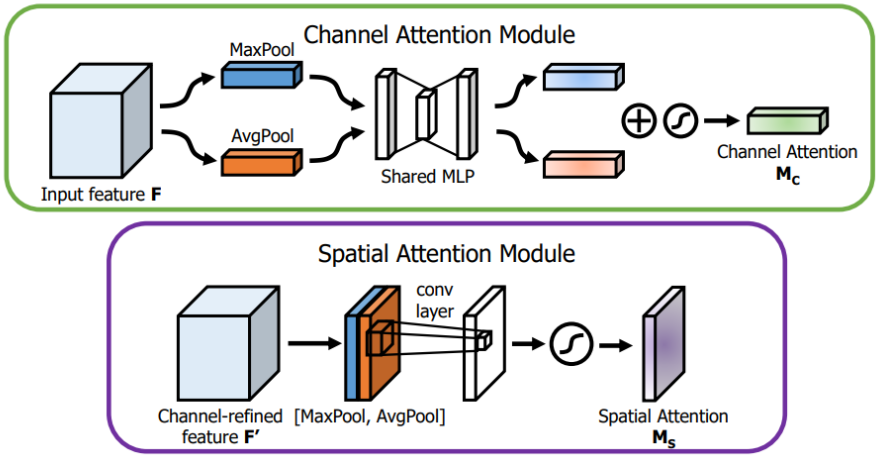


Figure 2.11: Diagram of CBAM attention modules [65]

2.2 Evaluation Metrics and Loss Functions

Loss functions and evaluation metrics occupy a critical position in the development and evaluation of artificial neural networks. These quantitative measures facilitate the determination of a model's performance and steer the optimization process throughout training. This section presents an overview of several most widely-used loss functions and evaluation metrics.

2.2.1 Mean Absolute Error

The Mean Absolute Error (MAE), alternatively referred to as L1-norm loss or L1 loss, serves as a loss function in machine learning and optimization contexts. This function measures the absolute distance between reconstructed image I' and original imageI'. Given an image size of $M \times N$, mathematically, this relationship can be expressed as follows[64]:

$$MAE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} |I(i,j) - I'(i,j)| \tag{2.2.1}$$

2.2.2 Mean Squared Error

Mean Squared Error (MSE), also known as L2-norm loss or L2 loss, is more sensitive to outliers and emphasizes larger errors due to the squaring operation. The L2 loss calculates the squared differences between the predicted values and the true values and then averages these squared differences over the entire dataset. Mathematically, it can be expressed as[51]:

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (I(i,j) - I'(i,j))^2 \tag{2.2.2}$$

2.2.3 Peak Signal-to-Noise Ratio

Peak Signal-to-Noise Ratio, also known as PSNR constitutes a prevalent metric in measuring the quality of reconstructed images, particularly in image compression and image denoising tasks. PSNR is a logarithmic measure that quantifies the difference between the original image and the reconstructed image, with higher PSNR values indicating better image quality. Mathematically, it can be formulated as[51]:

$$PSNR = 10 \cdot \log_{10} \frac{MAX_I^2}{MSE} \tag{2.2.3}$$

Where MAX_I^2 is the maximal of image data if the image is 8 bits, the possible maximal is $2^8 - 1$, which is 255. The unit for PSNR is in dB.

2.2.4 Structural Similarity Index Measure

The Structural Similarity Index Measure (SSIM) is another widely employed metric for evaluating the quality of reconstructed images. In contrast to PSNR, SSIM emphasizes the assessment of structural, luminance, and contrast information within images to more accurately reflect the human perception of image quality[51]. The SSIM index is calculated using a sliding window that traverses the entire image, comparing local regions within the window from both the reconstructed and original images. The SSIM can be mathematically expressed as:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.2.4}$$

where x and y are image windows from original image I and reconstructed image I'. μ_x and μ_y are their local means. σ_x^2 and σ_y^2 are local variance, σ_{xy} is the local cross-covariance between x and y. $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$ are small constants to prevent instability when the denominator is close to zero, where L is the dynamic range of the image, for 8-bit image, the L has the value of 255 which is $2^8 - 1$. the default value of k_1 and k_2 are 0.01 and 0.03. SSIM values range from -1 to 1, with a value of 1 signifying identical images and values approaching 1 denoting greater similarity.

2.2.5 Multi-Scale Structural Similarity

The Multi-Scale Structural Similarity (MS-SSIM) index is an extension of the Structural Similarity Index Measure (SSIM). Unlike SSIM, MS-SSIM provides a more comprehensive and robust measurement of perceptual image quality than SSIM.[60]. MS-SSIM works by applying SSIM calculations at multiple scales, which are obtained by iteratively down-sampling the images through a Gaussian pyramid. The final MS-SSIM score is computed as the product of the SSIM values at each scale, with the luminance, contrast, and structure components combined differently for each scale. The MS-SSIM can be mathematically expressed as:

$$\text{MS-SSIM}(x, y) = [l_M(x, y)]^{\alpha M} \prod_{j=1}^{M} [c_j(x, y)]^{\beta} [s_j(x, y)]^{\gamma} \tag{2.2.5}$$

where x and y represent the original and reconstructed image windows. M is the highest scale, $l_M(x, y)$ is the luminance comparison at the highest scale, $c_j(x, y)$ is the contrast comparison at scale j, and $s_j(x, y)$ is the structure comparison at scale j. The exponents α, β, and γ are weighting factors that control the relative importance of each component.

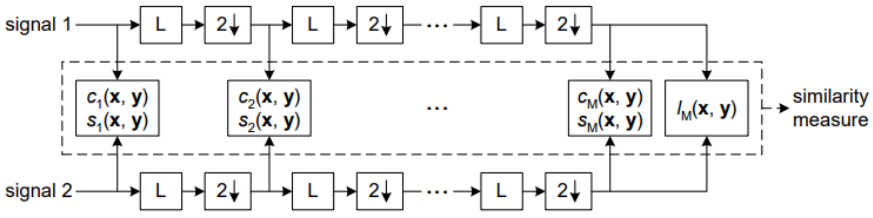


Figure 2.12: Diagrams of MS-SSIM measurement system, where L is low-pass filtering and $\downarrow$ is down-sampling by factor of 2 [60].

Chapter 3

Problem Definition and Related Work

Distinct from the existing obstruction removal problem, which aims to eliminate captured obstructions like fences or raindrops from images or videos. This thesis tackles the challenge of artifact removal in images taken through an obstruction-adhered glass. This problem can be commonly found in outdoor vision systems, particularly for surveillance camera systems. This problem was proposed by Dr. Xiaolin Wu, who is the supervisor of this work. To date, no feasible solution exists for such a problem, except works like Qian et al.[44], which focus on restoring clean images from raindrops adhered to a glass medium or surface of the camera lens. However, these works are limited to deraindrop, rather than removing various types of obstructions. Upon analyzing the artifacts caused by obstructions in the image dataset introduced in Chapter 5, this thesis categorizes the problem into three sub-problems: image deraindrop and image dehazing and small adherent particles removal.

3.1 Image Deraindrop

Images captured on a rainy day can experience significant degradation in quality and visibility, leading to poor performance in various visual applications such as surveillance systems and autonomous vehicles. As the result, image deraining is a notable challenge in the field of computer vision and image processing. In summary of existing work, most rain

Figure 3.1: An example of the image captured through a glass adhered with raindrops

models can be broadly categorized into three major groups: rain streak, raindrop, and a combination of rain and mist[53], while this work is mainly focused on raindrop removal. Given a clean background image B and the raindrop degradation artifacts in a tiny local coherent area D, the degraded image affected by Raindrop R_d can be modeled as:

$$R_d = (1 - M) \odot B + D \tag{3.1.1}$$

In the given equation, $\odot$ denotes the element-wise multiplication, while the binary mask M distinguishes if a pixel x is belonging to a raindrop region or the clean background. When $M(x) = 1$, pixel x is considered part of a raindrop region, or vice versa. Although water

droplets are essentially transparent, their shape and refractive index can cause pixels within a raindrop region to be influenced by the entire environment rather than a single point in the real world [69]. As a result, pixels within the raindrops' region appear to display unique imagery that is distinct from the background scene which is shown in Fig.3.1. Similar to the typical occlusion removal problem, when the camera is set to focus on the background scene, the imagery within the water droplets' region appears to be out of focus. However, certain areas of raindrops, particularly around the periphery and transparent regions, can still convey background information.[44]. Furthermore, this work assumes that other types of adherent obstructions exhibit behavior similar to raindrops, except that their light transmission rate is zero.

3.1.1 Multi-frame based approach

There are several approaches employ both spatial and temporal information to address the deraining problem. For instance, Garg and Nayar [15] suggest a method for detecting and removing rain from videos by exploiting the spatio-temporal characteristics of rain. They introduce a photometric model for rain, which captures the dynamic behavior of raindrops. By examining consecutive video frames, they calculate the average intensity of identified rain artifacts from preceding and following frames which will remove rain. However, their primary focus is on rain-strike removal. Alletto et al. [2] pioneered the application of learning-based methods for video raindrop removal tasks. They proposed a spatio-temporal generative adversarial network consisting of a self-supervised raindrop location estimator and a raindrop remover, wherein the spatial-temporal generator is jointly trained with optical flow. However, these methods necessitate multi-frame input, making them unsuitable for single-image deraindrop applications. Although Alletto et al. also

proposed a single-image raindrop removal model, its performance is not as promising as the spatio-temporal counterpart, and their method is trained with synthetic raindrop data.

3.1.2 Single image based approach

With the emergence of learning-based methods, the field of single-image raindrop removal has advanced rapidly. Eigen et al. [11] were the first to address the problem of water droplet removal with a learning-based approach, they develop a shallow convolutional network could restore the clean background B. Nonetheless, such a method has limitations due to the restricted performance of the CNN model. The resulting images exhibit artifacts and fakeness, particularly in regions with large and dense water droplets.

Qian et al.'s 2018 work, DeRaindrop [44], is possibly the first practical solution for raindrop removal using a generative adversarial (GAN) [18] approach. Their network includes an attentive recurrent network that incrementally generates an attention map, guided by the binary mask M from Eq. 3.1.1. To obtain the mask, Qian et al. simply subtract the raindrop-degraded image R_d from its corresponding clean image B and apply a threshold to determine whether a pixel is part of a raindrop region. In practice, they set the threshold to 30 for binary image masks in the training datset ant it is proves uffficeint for generating the attention map. They then feed the attention map into a contextual autoencoder to restore the background scene B. Additionally, they proposed a novel real-world deraindrop dataset for their method. Quan et al. [45] proposed using a double attention mechanism, incorporating a shape-driven attention mechanism and channel re-calibration, to simultaneously guide the CNN based on the dataset provided by Qian et al. The shape attention identifies raindrops by exploiting their physical properties, such as convexity and contour closedness. Furthermore, the channel re-calibration strategy is utilized to bolster the model's robustness

in handling raindrops of various shapes.

3.2 Image Dehazing

Figure 3.2: An example of image captured through a glass adhered with dirt

As shown in Fig. 3.3, In cases where the obstruction is small and thin, it decreases the visibility of the scene in a manner similar to fog, resulting in mist-like artifacts. To address this issue, He et al.[20] define a mist model for the task. Given R as a combination of obstructions such as raindrops, A as the scattering of adherent mist, and the clean background B, the degraded image I^{cap}, which suffers from mist artifacts, can be mathematically expressed as:

$$I^{cap} = ((1 - M) \odot B + R) \odot t + A \odot (1 - t) \tag{3.2.1}$$

In the adherent mist model, M represents the binary mask of obstructions, $\cdot$ denotes element-wise multiplication, and t is the transmission map, which indicates the information passing rate through the adherent mist[21]. Since there are the limited number of works primarily focused on adherent mist, this thesis expands the scope to several related studies on

atmospheric haze removal.

3.2.1 Traditional approach

Single image dehazing was initially investigated using prior-based methods, such as the dark channel prior proposed by He et al. [22]. In their work, He et al. assume that most local patches contain some pixels that exhibit very low intensities in at least one color channel in haze-free outdoor images. Based on this prior, the thickness of the haze can be determined to recover a high-quality haze-free image. However, this method often leads to incorrect color representation for mist-covered regions.

3.2.2 Learning-based approach

In recent years, learning-based methods have been proposed for the dehazing problem. Cai et al. [6] introduce an end-to-end CNN network for estimating transmission, incorporating a novel BReLU unit. Zhang et al. [70] present a deep learning-based approach for single-image dehazing using a densely connected pyramid dehazing network (DCPDN). The DCPDN combines both local and global context information and utilizes dense connections to facilitate the flow of information and gradients within the network. This network architecture enables the model to effectively learn haze removal from images while preserving image details.

3.3 Small Adherent Obstruction Removal

In contrast to image deraindrop and image dehazing, small adherent particles exhibit unique optical properties. The glass layer with adherent small particles imparts a certain radiance

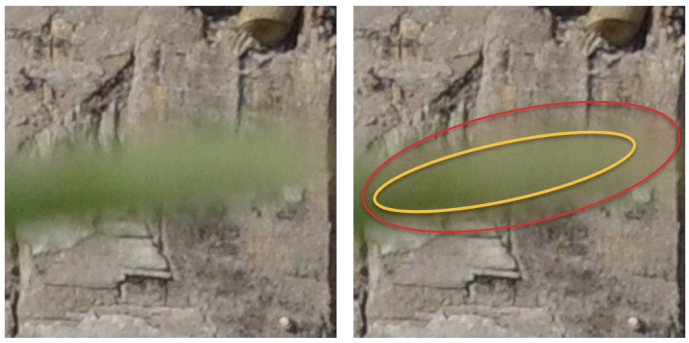


Figure 3.3: An example of small adherent particles can be observed in the following scenario: Within the red circle, obstruction artifact is semi-transparent, while within the yellow circle, obstruction artifact appear opaque.

to the camera by either scattering light from various environmental directions or reflecting light from its surface. Qiang Li, a member of our research team, proposed an adherent obstruction image formation model[36]. Given $I_s(x, y)$ as the radiance of the target scene, I as the image captured by the camera sensor, $\alpha(x, y) \in [0, 1]$ represents the attenuation ratio of the dirty glass layer, where 0 is for completely opaque and 1 is for entirely transparent, $I_o(x, y)$ signifies the intensification component which is the supplementary radiance from the dirty glass layer itself. As a result, the proposed image formation model can be depicted as:

$$I = \alpha \cdot I_s * h + (1 - \alpha) \cdot I_o * h \tag{3.3.1}$$

where $h(x, y, d)$ denotes the point spread function (PSF) at the defocus position d for the occlusion layer, and $*$ indicates image convolution. Consequently, small adherent particles removal can be viewed as a combination of defocus, deblurring and inpainting tasks.

3.4 Image Inpainting

Numerous deraindrop methods have drawn inspiration from image inpainting, such as incorporating GAN or cGAN networks into their deraindrop models to ”paint” visually pleasing results for raindrop-covered regions, as demonstrated in [44, 2]. Therefore, it is essential to mention the differences between image inpainting and image deraindrop. In contrast to image deraindrop, involves the restoration of damaged areas within an image while preserving overall coherence. In inpainting networks such as those found in [43, 19], an additional binary mask is provided as input to identify the damaged pixel regions. When target regions are provided, local assessments can be conducted to determine the realism of local areas. Therefore, the direct application of an inpainting algorithm is infeasible. Another related idea is image translation, like Pix2Pix [59]. Instead of directly restoring raindrop-covered pixels, it translates one image into another. This method proposes using a conditional GAN that learns not only the mapping between input and output images but also the loss function used to train the mapping. However, this technique represents a general mapping and is not specifically tailored for raindrop removal [44].

Chapter 4

A Dataset for Adherent Obstruction Removal

Data acquisition serves as a crucial element in the realm of machine learning endeavors, the efficacy of the trained models is inextricably linked to the precision, heterogeneity, and representativeness of the used data[27]. In this chapter, the author details the methodology of gathering and organizing real-world data from various sensors, as well as the generation of distinct categories of obstructions.

4.1 Sensors and Data Acquisition

During the data acquisition process, this work employs two cameras: a Sony A7RM2 full frame mirrorless camera with 85 mm GM lens and a Ricoh GR3X Advanced Photo System type-C (APS-C) camera with embedded 26.1 mm lens (Approx. 40mm in 35mm equivalent focal length). In this study, a set of image pairs is needed. Each image pair comprises the same background scene, with one image degraded by obstructions and the other free

of obstructions. To address this concern, this work employs two pieces of identical clean camera glass filters, positioning them in front of the camera interchangeably(both have a light transmittance rate of 98.4% and a thickness of 1 mm). One filter is covered with obstructions, while the other is maintained in a clean state. This approach ensures consistency in the refractive properties between the obstruction-adhered glass filter and the clean glass filter, thereby minimizing the potential for pixel misalignment[44].In an effort to further minimize pixel misalignment between the ground truth and degraded images, utilizes a standard tripod equipped with remote control for stabilizing the cameras during data collection. However, misalignments induced by wind or moving objects, such as shaking leaves or walking pedestrians may still transpire. To mitigate the impact of these disturbances, the author performs a manual selection process upon the completion of the data acquisition stage, ensuring that the final dataset is free from misalignment introduced by any movement-related factors. In order to enrich the diversity of the dataset, the distance between a glass filter and the camera is varied from 3-12 cm[1].

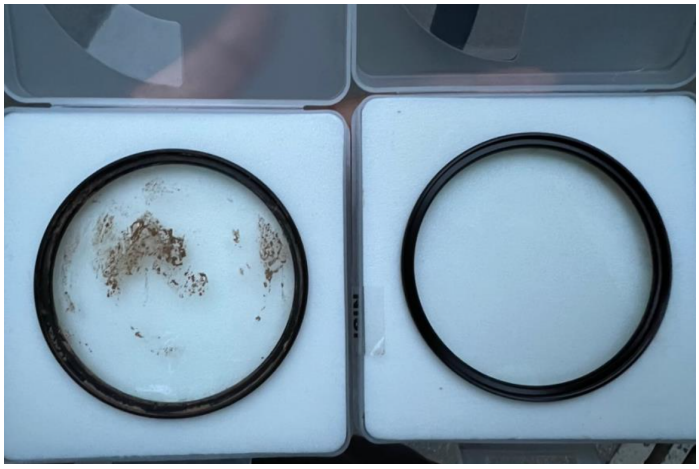


Figure 4.1: Glass filter adhered by dirt(left). Clean glass filter (right).

[1]The distance between a glass filter and the camera is defined as the distance from the glass to the surface of the camera lens, rather than the distance to the center of the camera sensor.

Since cloud coverage can introduce variations in scene brightness, atmospheric conditions are also taken into account as a constraint. To maintain consistency, all data is collected under constant ambient conditions and against still backgrounds. Additionally, the author meticulously cleans both the lens and camera sensor prior to each data collection process in order to prevent any undesired dust particles from appearing on either the camera sensors or the lens.

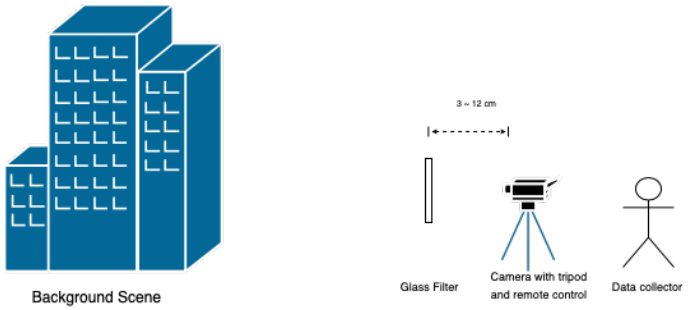


Figure 4.2: An illustration of equipment setup process

4.2 Choice of Obstructions

The dataset includes the four most common types of obstructions that are encountered in outdoor computer vision systems: clear raindrops, muddy raindrops, dirt, and tiny particles. To simulate clear raindrops, the author sprays tap water onto the glass surface. Muddy raindrops are replicated by spraying a mixture of water and mud. Dirt is represented using clay and moistened sand adhered to the glass. For tiny particles, combinations of small debris are obtained, including cut leaf particles, flower particles, stone fragments, and hand-drawn scratches with a marker pen.

Figure 4.3: From left to right: raindrop sample, muddy raindrop sample, dirt sample, and tiny particle sample

4.3 Data Patch and Selection

Image pairs taken from the Sony A7RM2 camera have dimensions of 7952 × 5304, while those collected from the Ricoh GR3X camera have dimensions of 3360 × 2240. Due to the limitation of the GPU memory, utilizing such large image files directly into GPU as training data is impractical and computationally expensive[71]. Therefore, for each data pair, the degraded image is cropped into many 256x256 patches, as well as its corresponding obstruction-free ground truth. To accomplish this, this work develops a Python script utilizing the OpenCV library, which crops the images based on mouse-clicked coordinates. Additionally, to better visualize the pixel misalignment, a binary mask is obtained by performing subtraction between the degraded image and its associated ground truth. In cases where the misalignment is visibly substantial, the image pair is discarded and subsequently excluded from the dataset.

As the result, the training dataset consists of 2,637 image patches and the testing dataset consists 333 image patches.

Figure 4.4: **A comprehensive illustration of the image selection process.**(white boxes denote pixel misalignments introduced by wind-induced disturbances). From left to right: Degraded image, ground truth, binary subtraction mask with apply with a heated map

Figure 4.5: An illustration of the image cropping process: Green box indicated the crop patch

Chapter 5

Development of Attentive Multi-Branch Encoder-Decoder Network

5.1 Baseline Model

Encoder-decoder networks have proven effective in image restoration tasks, owing to their capability to learn compact data representations and their proficiency in reconstructing input data from these representations[67, 7]. To validate the encoder-decoder networks's ability for occlusion removal, this work initially implements a simple coarse U-shaped network for the deraindrop problem, utilizing the dataset proposed by [44]. Consequently, the images recovered using the U-net exhibit a discernible improvement in the PSNR metric and at the perceptual level.

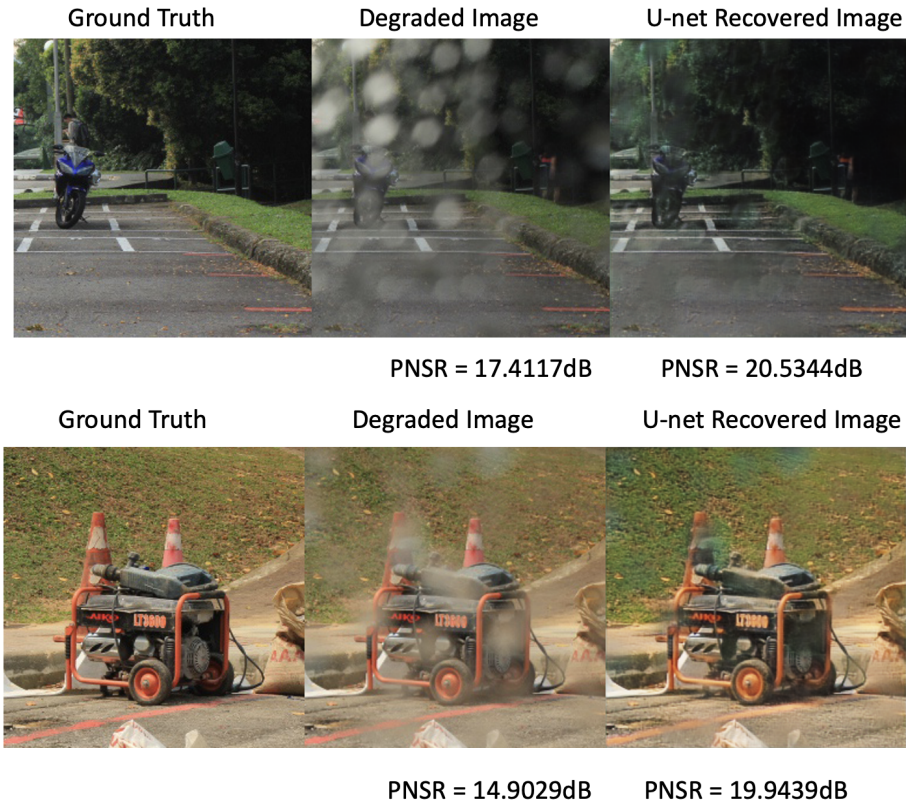


Figure 5.1: A comparison between the ground truth image, the rain-affected image, and a reconstructed image obtained using a coarse U-shaped network.

5.2 Attentive Multi-Branch Encoder-Decoder Network

In this section, the study introduces a comprehensive network incorporating attention mechanisms and test-time adaptation for occlusion removal problems. Inspired by the network proposed by Chi et al.[7] which leverages auxiliary learning to achieve the function of test-time adaptation. The proposed framework also encompasses two tasks: the primary task and the auxiliary task. Given an obstruction-degraded image I_d, the objective is to restore an obstruction-free clean image. The primary task accepts degraded images containing occlusions (denoted as I_d) as input and endeavors to predict a corresponding clean image I_c.

The auxiliary branch synthesizes occlusion artifacts to augment the primary task's performance. Simultaneously, the auxiliary task is tailored to learn specific occlusion features for each input, thereby allowing the model parameters to adapt to individual test images and subsequently supporting the primary task, resulting in superior outcomes[56, 7]. Both tasks predominantly share the model's structure, which constitutes an encoder-decoder network.

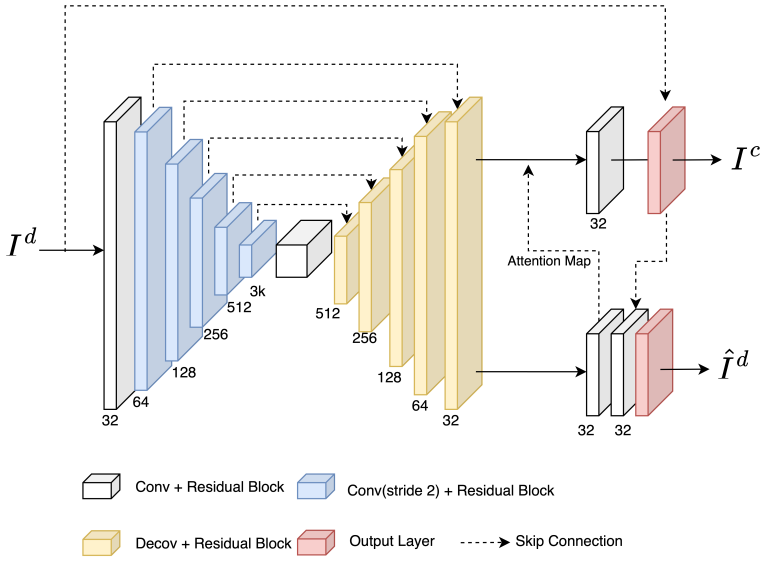


Figure 5.2: An illustration of proposed obstruction removal network

5.2.1 Primary Occlusion Removal Network

The primary occlusion removal network proposed in this study takes a degraded image I^d as input and outputs a corresponding clean image I^c. This network comprises three main components: convolution blocks, deconvolution blocks, and residual blocks. Convolution blocks, characterized by a stride of 2, serve as feature extractors for the degraded

image, capturing essential information such as sharp edges, structure, and patterns, subsequently down-sampling and compressing these elements into the latent space. Deconvolution blocks function to enhance the resolution of feature maps, striving to reconstruct data from its compressed representation. The core idea behind devising such deconvolution blocks is that, during the devolution process, the model can be trained to ignore obstructions and generate plausible content for corrupted areas, resulting in obstruction-free images. Residual blocks, implemented following both convolution and deconvolution blocks, assist in mitigating the vanishing gradient problem and augmenting the network's overall performance. In instances where occlusions may not be entirely removed after the decoder-encoder network, a convolution block with 0 stride is employed to enhance the output of a visually appealing clean image.

In contrast to the architecture proposed by Chi et al.[7] for deblurring scenes, restoring obstruction-polluted images requires the network's ability to precisely identify the region of obstruction artifacts within the degraded image patch. Therefore, this study incorporates the concept of attention mechanisms[57, 65]. An attention map, activated by a sigmoid function, is derived from the auxiliary branch and applied as a dot product to the primary network, as shown in Fig. 5.2. This attention map serves as a guide to the primary network in enhancing the visual representation of regions previously obscured by obstruction artifacts.

5.2.2 Self-Supervised Auxiliary Network

Considering the infeasibility of including every possible types of obstruction in the dataset, the primary intuition behind implementing an auxiliary branch is that, the auxiliary task enables the network to adjust internal parameters when test data differs from the training

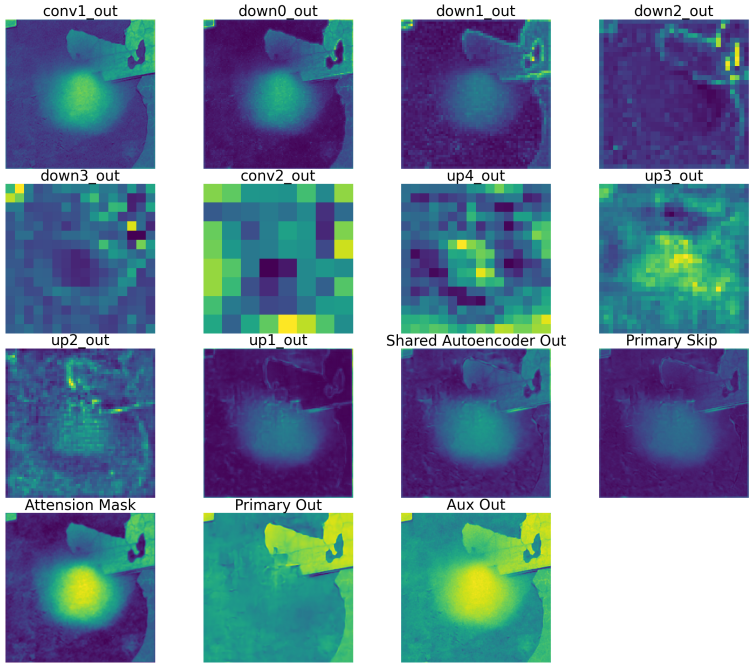


Figure 5.3: **An illustration of feature maps at each stage.** The attention mask accentuates the region containing occlusion artifacts, while the additional convolution block improves the visual representation of the result for the primary task. Concurrently, the auxiliary task generates a synthetic version of obstruction artifacts.

data distribution, also known as test-time adaptation[13, 8, 55]. Similar to the approach by Chi et al., this work also chooses to generate a synthetic version of degradation as the auxiliary task, incorporating the residual information from the output of the primary task (As shown in Fig. 5.2, the residual information is transmitted through skip connections from the primary branch to auxiliary branch). As the loss calculation for the auxiliary task solely relies on the comparison between the input degraded image and the synthetic degraded image, this process facilitates the optimization of the pre-trained model in a way that allows rapid adaptation to any test image, even those not previously encountered.

Furthermore, during the test phase, the model is able to obtain specific details about the obstruction artifacts, such as their region. This information can be employed as an

attention map incorporated into the main occlusion removal task. the performance of the primary task is significantly improved in terms of both quantitative metrics and visual representation.

5.2.3 Model Optimization

Loss Function

Selecting a suitable loss function is crucial for optimizing a model effectively. After thorough investigation, this study employs the MS-SSIM L1 loss function introduced by Zhao et al. [72], which is an amalgamation of the L1 loss and the Multi-Scale Structural Similarity (MS-SSIM) index. This hybrid loss demonstrates a superior performance when applied as a regularizer in an image restoration network, compared with other losses, such as the widely-used l_2 loss. The MS-SSIM L1 loss merges the benefits of both pixel-level sensitivity and perceptual similarity. Consequently, this fosters a neural network's capacity to generate outputs that exhibit not only precise pixel values but also visual appeal. Given the primary task as P and the auxiliary task as A, mathematically, MS-SSIM L1 loss for the primary task and the auxiliary task can be expressed as:

$$\mathcal{L}_P = \alpha_0 \cdot \mathcal{L}_P^{\text{MS-SSIM}} + (1 - \alpha_0) \cdot G \cdot \mathcal{L}_P^{\ell_1} \tag{5.2.1}$$

$$\mathcal{L}_A = \alpha_0 \cdot \mathcal{L}_A^{\text{MS-SSIM}} + (1 - \alpha_0) \cdot G \cdot \mathcal{L}_A^{\ell_1} \tag{5.2.2}$$

where G denotes the Gaussian weights and the subscripts P and A denote the primary and auxiliary task. The $\alpha_0 = 0.84$ is set as default.

Network Training

Let $\theta = \theta_s, \theta_p, \theta_a$ represent the parameters of the entire network, where θ_s denotes the shared parameters of the encoder-decoder network, θ_p signifies the parameters of the primary branch, and θ_a corresponds to the auxiliary branch parameters. Let f_p^θ and f_a^θ represent the functions of the primary branch and the auxiliary branch, respectively. The predicted obstruction-free image can be expressed as:

$$I^c = f_p^\theta(I^d; \theta_s, \theta_p, \theta_a) \tag{5.2.3}$$

And the synthetic degraded image can be expressed as:

$$\hat{I^d} = f_a^\theta(I^d; \theta_s, \theta_p, \theta_a). \tag{5.2.4}$$

Due to the limitations of the hardware, this study only employs offline training for the neural network. The total loss is obtained through the combination of the primary loss and the auxiliary loss:

$$\mathcal{L}_{total} = \beta \cdot \mathcal{L}_P(I^c, \hat{I}^c; \theta_s, \theta_p, \theta_a) + (1 - \beta) \cdot \mathcal{L}_A(I^d, \hat{I}^d; \theta_s, \theta_p, \theta_a), \tag{5.2.5}$$

where β is set to 0.8 in this work. Thereby, the main objective of the network can be defined as:

$$\min_{\Theta^s, \Theta^p, \Theta^a} \sum_{k=1}^{K} \mathcal{L}_{total}^{(k)}(I^d, \hat{I}^d, I^c; \theta_s, \theta_p, \theta_a) \tag{5.2.6}$$

Test-time Adaptation

During the testing phase, by applying auxiliary loss from Eq.5.2.2, the adapted parameter $\tilde{\theta}_s$ and $\tilde{\theta}_a$ can be obtained. Subsequently, $\tilde{\theta}_s$ is utilized in the shared encoder-decoder architecture to augment the performance of the primary task when handling previously unseen data with a distinct distribution from the training data.

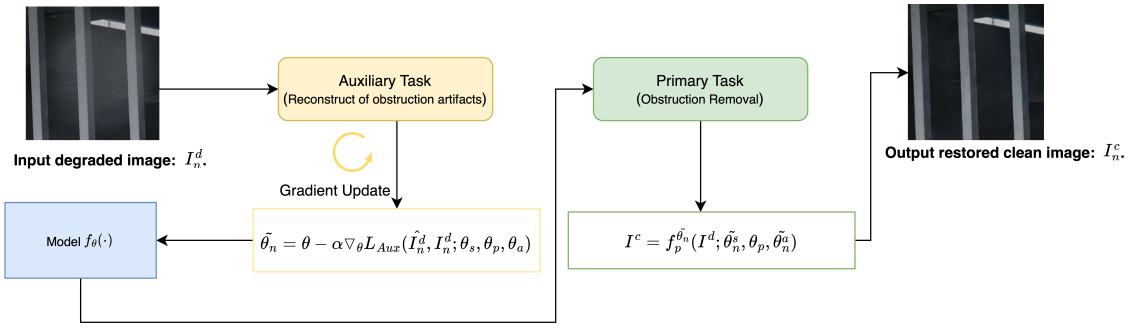


Figure 5.4: **An illustration of test-time adaptation process.** During the test-time adaptation process, the auxiliary task updates the shared parameters using Eq. 5.2.2. Following this update, the model conducts obstruction removal on the primary branch.

Catastrophic forgetting

Due to hardware limitations, this study exclusively employs offline training for the proposed network. However, the primary and auxiliary tasks are two unconnected tasks. The weight updates through the auxiliary loss are more biased towards enhancing the synthetic degraded image quality, rather than the quality of the obstruction-free image. This undesired effect is also referred to as catastrophic forgetting [7]. To address this issue, future improvements may consider, such as implementing meta-auxiliary learning (MAXL) scheme [35] alongside a more advanced GPU. This approach trains both the main and auxiliary tasks separately and enforces a constraint that parameter updates via the auxiliary loss should positively impact the primary task.

Chapter 6

Preliminary Experiments

In this chapter, extensive experiments are performed on the proposed method to validate its effectiveness and demonstrate the robustness of the approach.

6.1 Datasets and Metrics

The neural network is trained with the dataset introduced in Chapter 4. The proposed dataset comprises 2637 image patches, while the testing dataset consists of 333 image patches. Each image patch has a resolution of 256×256. In this section, the evaluation metrics employed are PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index Measure).

6.2 Implementation and Training Details

For all implementations, Python is utilized as the programming language and PyTorch is employed as the framework. The network is trained for 1,300 iterations, optimized by the

Adam optimizer [28] with β_1 set to 0.5 and β_2 set to 0.999. The learning rate and batch size are set at 0.0001 and 30, respectively. During the test-time adaptation, 6 gradient updates are performed for each test image, with the learning rate for gradient updates fixed at 0.000006. All pixel values for both training and testing images are scaled to the range [-1, 1], and the LeakyReLU activation function [68] with a slope of 0.1 is used. Network training, testing, and adaptation are executed on an Nvidia RTX 3090 GPU[1].

6.2.1 Method Comparison

An objective and subjective quality comparison is carried out on the proposed real-world dataset. As most existing approaches focus solely on specific types of obstructions, such as raindrops, and no articles address the removal of adherent various obstruction artifacts from images taken through glass. As this research is conducted using a novel dataset, it necessitates retraining other approaches to ensure a fair comparison. However, some deraindrop methods discussed in Chapter 3 do not supply training codes for their respective networks. Therefore, this work compares the proposed network with several prominent methods with provided open-source code, each specializing in a different area: PReNet for image deraining [46], U-net for image deraindrop [66], DeblurGAN-v2 for image dehazing and deraindrop [32], and CTSDG for image inpainting [19]. Besides that, the implementation of this work is built upon Pytorch framework, U-net is initially proposed for medical image segmentation. Since it incorporates an encoder-decoder network, it is capable of removing raindrops during the upsampling stage, as discussed in Section 5.1. Similarly, DeblurGAN-V2, originally designed for blind motion removal, utilized a Feature Pyramid Network in both the generator and discriminator. The Feature Pyramid Network within the

[1] https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090-3090ti/

Methods \ Dataset	OROS	
	PSNR	SSIM
Degraded	23.31	0.846
PReNet [46]	22.80	0.835
U-net [66]	24.70	0.846
DeblurGAN-v2 [32]	28.18	0.857
CTSDG [19]	24.96	0.794
Ours	**29.91**	**0.879**

Table 6.1: **Comparison of the proposed model with prominent open-source methods.**All the networks are retrained using the obstruction dataset. The proposed network demonstrates a superior performance in comparison to others.

generator allows the model to capture multi-scale features and reconstruct deblurred images at various resolutions, resulting in sharp and realistic deblurred images. Consequently, we retrained this method to assess the potential of a combination of GAN and Feature Pyramid Network. In this study, binary masks are computed by subtracting the degraded adherent obstruction image from a clean ground truth image. These masks are subsequently utilized for retraining and evaluating the CTSG inpainting model. For each of these methods, the network is retrained using the officially released training code and the proposed occlusion dataset from Chapter 4. The hyperparameters for each retraining process are set to default values. Consequently, the comparison remains fair and faithful across various datasets.

6.2.2 Quantitative Evaluation

The objective performance using PSNR and SSIM as evaluation metrics on the proposed dataset, is presented in Table 6.1. This work also examines the performance on each distinct type of obstruction, as displayed in Table 6.2. The results reveal that the proposed method in this study achieves the best PSNR and SSIM performance on the obstruction dataset,

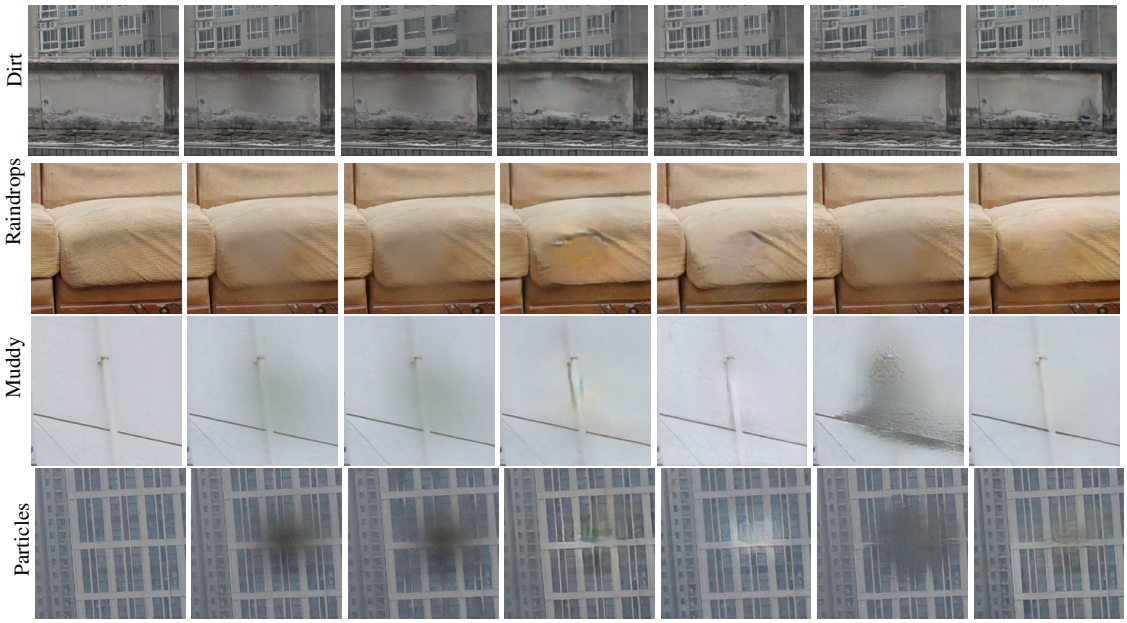
Methods \ Dataset	Dirt		Raindrops		Muddy Water		Particles	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Degraded	25.79	0.861	24.83	0.827	24.25	0.833	23.07	0.788
PReNet[46]	25.10	0.850	24.52	0.817	23.51	0.809	19.62	0.836
U-net[66]	24.19	0.834	24.73	0.826	25.48	0.827	24.92	0.872
DeblurGAN-v2[32]	28.52	0.856	28.28	0.832	27.12	0.835	28.15	0.877
CTSDG [19]	26.38	0.805	26.22	**0.872**	24.94	0.758	23.07	0.788
Ours	**29.93**	**0.878**	**29.86**	0.852	**29.53**	**0.852**	**30.18**	**0.899**

Table 6.2: **A comparison is conducted on the proposed dataset, with four categories: dirt, raindrops, muddy raindrop, and particles**. The network presented in this study demonstrates a superior performance compared to other approaches.

with the exception of CTSDG for Raindrop sub-dataset. Figure 6.4 illustrates the effect of implementing auxiliary task for test-time adaptation. After four gradient updates (test-time adaptation), there is a 0.4 dB enhancement in PSNR and a 6e-3 improvement in SSIM. The impact of catastrophic forgetting begins to manifest after 4 steps of gradient update, leading to a decline in overall performance for both PSNR and SSIM metrics.

6.2.3 Qualitative Evaluation

This section presents a comparison of the objective (visual) results of the proposed network with other different approaches, as shown in Fig. 6.1. Upon close examination, the method proposed in this work demonstrates a more visually pleasing result. For the dirt example, the proposed method produces fewer pattern artifacts near the surface of the wall, while other approaches like DeblurGAN-V2 and U-net[66] yield false colors and strange patterns. For particles and muddy raindrop artifacts, the proposed method yields finer details on edges and regions covered by obstructions. CTSDG[19] introduces significant artifacts in the muddy raindrop and particles obstruction sub-datasets. This could be attributed to the noise introduced by the camera sensor. Training the CTSDG network necessitates a mask of



(a) GT (b) Input (c) PReNet (d) U-net (e) DdlrGAN2 (f) CTSDG (g) Ours

Figure 6.1: **Qualitative comparison with different approaches on proposed dataset.** The proposed method generates results with fewer artifacts and greater visual appeal compared to alternative approaches (where DblrGAN2 stands for DeblurGAN-V2)

the corrupted area as input; however, the mask provided by subtracting the ground truth and obstruction-degraded image contains visible sensor noise, which could potentially affect the performance of the CTSDG network. PReNet [46] exhibits suboptimal performance as a deraining network, particularly on the Raindrops sub-dataset.

6.2.4 Evaluation on Unseen Obstruction Types

As discussed in Chapter 5, test-time adaptation can improve performance on unseen data which has a distinct distribution from the training data. To verify this, this work retrains the proposed network, CTSDG, and DeblurGAN-v2 using a training dataset consisting of only three categories: Muddy raindrops, raindrops, and particles. Subsequently, all approaches are evaluated on a dirt-degraded test dataset, where dirt obstruction artifacts

Methods \ Dirt	Unseen Ones	
	PSNR	SSIM
DeblurGAN-v2[32]	25.57	0.802
CTSDG [19]	25.29	0.814
Ours (no update)	29.39	0.871
Ours (3 updates)	**29.67**	**0.876**

Table 6.3: **Evaluation on Unseen Obstruction Types.**

exhibit a different distribution from the other three types of degradation. As shown in Table 6.3, the test-time adaptation yields a 0.3 dB enhancement in PSNR and a 0.05 improvement in SSIM metrics. The performance of the proposed method surpasses that of the other two approaches.

6.2.5 Computational Cost

This work evaluates the execution speed of processing a 256×256 image for the proposed network on an RTX 3090 GPU. The proposed method requires 0.17s without test-time adaptation and 1.30s with 6 gradient updates. In contrast, DeblurGAN-v2 necessitates 1.68s due to the need for additional features from a pre-trained network. Through comparison, our method proves to be simple yet efficient, even when incorporating test-time adaptation process.

6.2.6 Ablation Studies

This section provides an additional ablation experiment on the proposed network, focusing on each individual components and their impacts. Additionally, the influence of the number of gradient updates during test-time adaptation is examined to gain further insights into the network's performance.

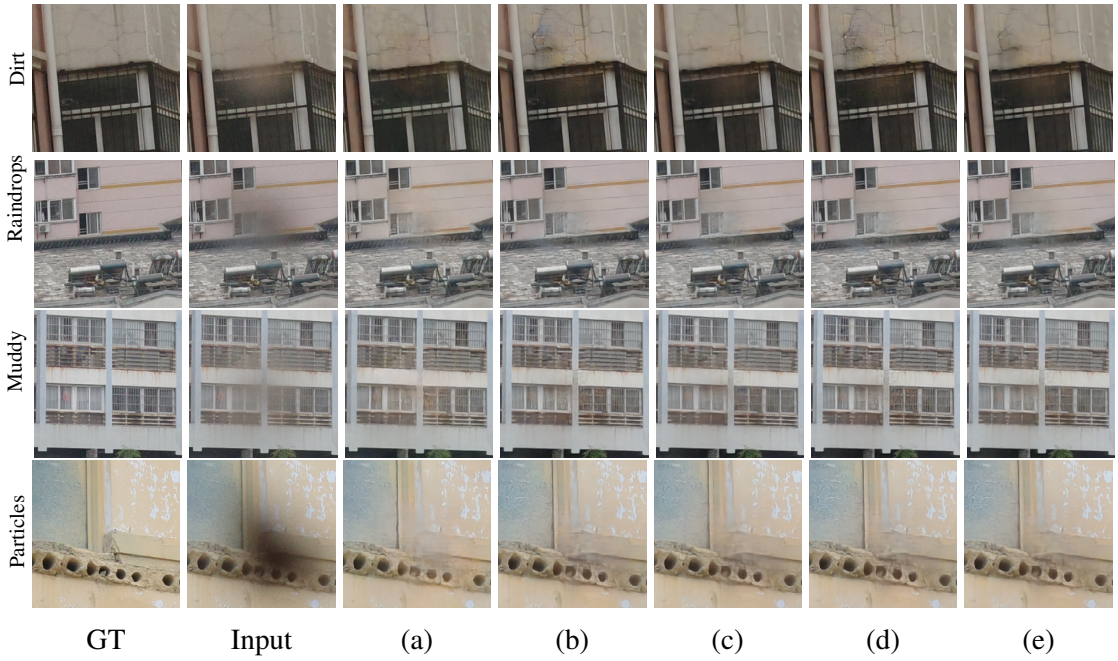


Figure 6.2: **Qualitative comparison with different network structures.** (a) Primary; (b) Primary + Auxiliary; (c) Primary + Auxiliary + Update; (d) Primary + Auxiliary + Attention; (e) Primary + Auxiliary + Update + Attention.

Network Structures	PSNR	SSIM
Primary	29.16	0.869
Primary + Auxiliary	29.42	0.872
Primary + Auxiliary + Update	29.82	0.875
Primary + Auxiliary + Mask	29.50	0.873
Primary + Auxiliary + Update + Mask	**29.91**	**0.879**

Table 6.4: **Ablation studies on network structures.** Including the auxiliary-learning through self occlusion reconstruction enhance the performance of primary task. As gradient updates and the attention mask are utilized, the performance of the primary task improves more.

Network Structures

This work assesses and retrains each component of the proposed network to investigate their respective impacts: (a) The network comprises solely the primary branch; (b) The network incorporates both primary and auxiliary branches; (c) The network consists of both primary and auxiliary branches and is updated through test-time adaptation; (d) The network encompasses both primary and auxiliary branches, as well as an attention mechanism. (e) The network comprises both primary and auxiliary branches, is updated through test-time adaptation, and incorporates an attention mechanism. In the ablation studies, the number of gradient updates is set to 4. Table 6.4 showcases the quantitative comparison of networks with different components, trained on the proposed dataset. Cases (b) and (d) demonstrate the effectiveness of utilizing the auxiliary task and attention mechanism in improving the performance of the primary task, while case (e) exhibits a noticeable increase in performance with the help of test-time adaptation. The qualitative evaluation depicted in Fig. 6.2 also highlights the effectiveness of implementing the auxiliary task, test-time adaptation, and attention mechanism.

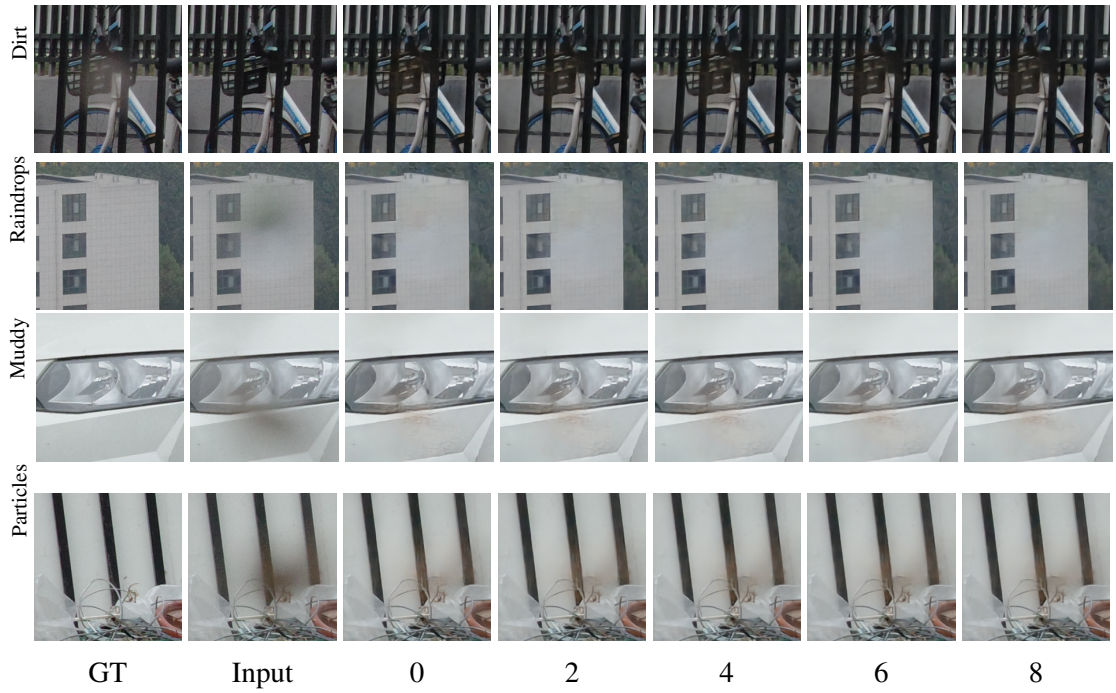


Figure 6.3: Qualitative examples are provided to demonstrate the improvement of test-time adaptation on a real dataset.

Evaluation on test-time adaptation

This subsection highlights the impact of test-time adaptation. During the testing phase, the network performs $N = 1, 2, 3, ..., 10$ gradient updates on the pre-trained model. A qualitative comparison performed on particles, muddy raindrops, raindrops, and dirt sub-dataset is shown in Fig.6.3, while the numerical evaluation of test-time adaptation is illustrated in Fig.6.4. Both figures corroborate the advantages of implementing test-time adaptation. When N reaches 4, the output obstruction-free image attains the highest PSNR value. However, as the number of gradient updates increases, the catastrophic forgetting effect begins to manifest, leading to a decline in both PSNR and SSIM metrics and the appearance of more visible artifacts.

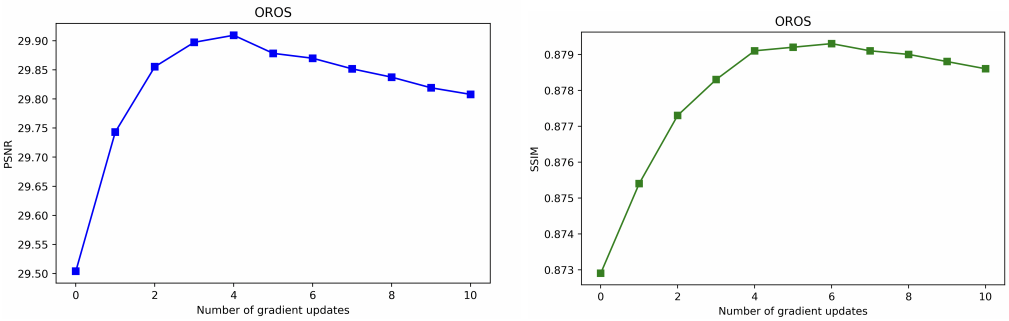


Figure 6.4: **Illustration of PSNR and SSIM after each gradient update.** Without test-time adaptation, the pre-trained model initially performs worse. Model is able to grain performance after applying gradient update. The peak PSNR is achieved after 4 times gradient update

Chapter 7

Conclusion and Future Work

This chapter provides a summary of the key findings and insights gained throughout the development of this thesis. Moreover, this chapter also delves into potential improvements for the proposed method, exploring opportunities for further refinement and enhancement.

7.1 Conclusion

This thesis tackles a complex problem, namely the single image adherent obstruction removal. Given an image captured through a glass with adhered obstructions, the objective is to eliminate all artifacts caused by these obstructions. To gain a deeper understanding of the problem, this thesis begins with an examination of the conceptual theory of popular deep learning methods such as CNNs and autoencoders. This paper also reviews several prominent approaches to addressing the obstruction removal problem.

Furthermore, this work presents a test-time adaptation network that incorporates auxiliary learning and an attention mechanism. During the testing phase, the auxiliary branch updates the shared internal-parameters of shared encoder-decoder network, enabling it to

restore images not only from known obstruction categories but also from unseen ones. Regarding the network's application, adherent occluders tend to persist on the glass surface for extended periods. Consequently, in real-world applications such as surveillance cameras that utilize video outputs, test-time adaptation only needs to be applied occasionally. Designing a spatial-temporal model would be unnecessary and would increase the network's complexity.

To better represent real-world scenarios, this work introduces a new dataset covering a wide range of common real-world obstructions along with their corresponding clean ground truth images. Experimental results demonstrate that the proposed method outperforms alternative approaches in both quantitative and qualitative evaluations.

7.2 Future Work

There are several limitations to this work. One major issue is catastrophic forgetting which has been discussed in Chapter 5. Chi et al.[7] suggested solving this problem by adopting a meta-training technique during the model optimization process. However, due to hardware limitations, implementing such a technique requires a GPU with significantly larger memory, whereas the RTX3090 only has a memory size of 24GB. Moreover, the qualitative experiment indicates that visible artifacts still persist in the restored obstruction-free images. This might be attributed to the fact that the proposed network only has one convolutional block in the primary branch. A more powerful design for this branch can be considered for future improvements. For example, the proposed method could be redesigned following the coarse-to-fine network approach proposed by Liu et al [37], in which the shared encoder-decoder structure produces a coarse version of the restored obstruction-free image, while another encoder-decoder network is implemented as the primary branch to refine the output

quality. Additionally, the idea of implementing a Generative and adversarial network like DeblurGAN-v2[32] could also be considered for future enhancements.

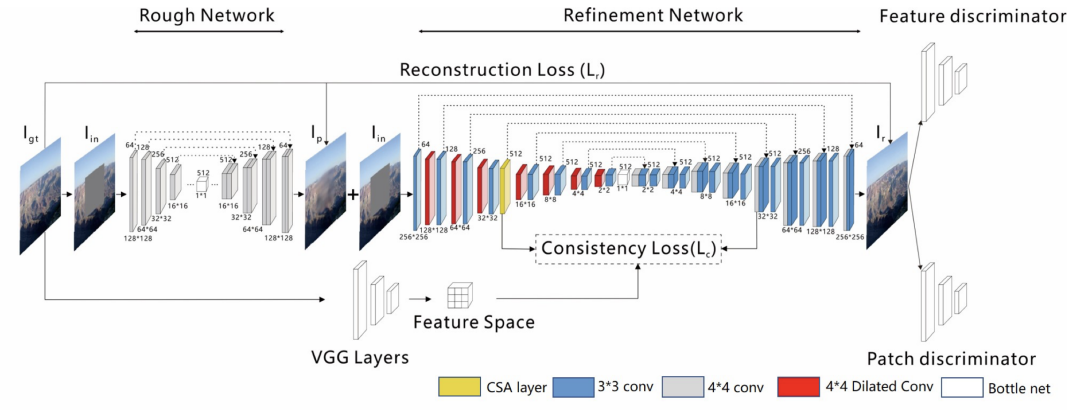


Figure 7.1: Network architecture proposed in Coherent Semantic Attention for Image Inpainting)

Appendix A

Python Implementation for Image Cropping

```python
import numpy as np
import cv2
import sys
import os

# image point
p2d = np.array( [0, 0] )

# crop size
crop_x = 256 #128
crop_y = 256


def onMouse(event, x, y, flags, param:np.array):
    if event == cv2.EVENT_LBUTTONDOWN:
        global p2d
        p2d[0] = x
        p2d[1] = y
        img = param.copy()
        cv2.rectangle(img,(x-crop_x,y-crop_y),(x+crop_x,y+crop_y),(0,255,0),2)
        cv2.imshow('img', img)

def saveImg(degrade,gt,cate):
    # degrade_path   = 'mytrain2/'+cate+'/Degraded'
    # gt_path    = 'mytrain2/'+cate+'/GT'
    degrade_path = 'Validate/Degraded'
    gt_path = 'Validate/GT'
    if not os.path.exists(degrade_path):
        os.makedirs(degrade_path)
    if not os.path.exists(gt_path):
        os.makedirs(gt_path)

    uniq = 0
    save_deg= os.path.join(degrade_path, '0.png')
    save_gt = os.path.join(gt_path, '0.png' )
#Overwrite if missing associating pair gt/degraded image
    while (os.path.exists(save_deg) and os.path.exists(save_gt)): #bug need to be fixed
        uniq += 1
        save_deg= os.path.join(degrade_path, '0%s.png' % (uniq))
        save_gt = os.path.join(gt_path, '0%s.png' % (uniq))

    print("Saving..\n", save_deg,save_gt)
    cv2.imwrite(save_deg,degrade)
    cv2.imwrite(save_gt,gt)

def saveImg2(degrade,gt,cate):
    degrade_path   = 'test'+'/Degraded'
    gt_path    = 'test'+'/GT'
    if not os.path.exists(degrade_path):
        os.makedirs(degrade_path)
    if not os.path.exists(gt_path):
        os.makedirs(gt_path)

    uniq = 0
    save_deg= os.path.join(degrade_path, '0.png')
    save_gt = os.path.join(gt_path, '0.png' )
#Overwrite if missing associating pair gt/degraded image
    while (os.path.exists(save_deg) and os.path.exists(save_gt)):
        uniq += 1
        save_deg= os.path.join(degrade_path, '0%s.png' % (uniq))
        save_gt = os.path.join(gt_path, '0%s.png' % (uniq))

    print("Saving..\n", save_deg,save_gt)
    cv2.imwrite(save_deg,degrade)
    cv2.imwrite(save_gt,gt)

#Click to choose center of square (default 300x300)
#Pressing "S" to save patch
#Pressing q to quit current state
#Pressing esc to terminate program
```

```
72  if __name__ == '__main__':
73
74      #Create two subfolder to store GT & Degraded image
75      #Categ = "Raindrop"
76      patch_num = "Fourth_Patch/"
77      Categ = "Dirt"
78      degrade_path   = patch_num+ '/Degraded'
79      gt_path    =patch_num + '/GT'
80      for name in sorted(os.listdir(degrade_path)):
81          if not name.startswith('.'):
82              degrade_file = os.path.join(degrade_path, name)
83              gt_file = os.path.join(gt_path, name)
84
85              degrade_img = cv2.imread(degrade_file)
86              gt_img = cv2.imread(gt_file)
87              print("\n Accessing:",degrade_file, gt_file)
88
89              while True:
90                  cv2.imshow('img', degrade_img)
91                  cv2.setMouseCallback('img', onMouse, param = degrade_img)
92                  k = cv2.waitKey(0)
93                  if k == ord('s'):
94                      crop_deg = degrade_img[p2d[1]-crop_y:p2d[1]+crop_y, p2d[0]-crop_x:p2d[0]+crop_x]
95                      crop_gt = gt_img[p2d[1]-crop_y:p2d[1]+crop_y, p2d[0]-crop_x:p2d[0]+crop_x]
96                      saveImg(crop_deg,crop_gt,Categ)
97                  elif k== ord('q'):
98                      break
99                  elif k == ord('='): #Exit on =
100                     sys.exit()
```

Appendix B

Python Implementation for Image Selecting

```python
import numpy as np
import cv2
import os
import sys


def saveImg(degrade,gt):
    degrade_path   = 'First_Patch/Raindrop/Degraded'
    gt_path    = 'First_Patch/Raindrop/GT'
    if not os.path.exists(degrade_path):
        os.makedirs(degrade_path)
    if not os.path.exists(gt_path):
        os.makedirs(gt_path)

    uniq = 0
    save_deg= os.path.join(degrade_path, '0.png')
    save_gt = os.path.join(gt_path, '0.png' )
#Overwrite if missing associating pair gt/degraded image
    while (os.path.exists(save_deg) and os.path.exists(save_gt)):
        uniq += 1
        save_deg= os.path.join(degrade_path, '0%s.png' % (uniq))
        save_gt = os.path.join(gt_path, '0%s.png' % (uniq))

    print("Saving..\n", save_deg,save_gt)
    cv2.imwrite(save_deg,degrade)
    cv2.imwrite(save_gt,gt)

if __name__ == '__main__':
    categ = 'Crop/Crop_Dirt'
    #categ = 'final_data/Test/'
    raindrop   = categ+'/Degraded'
    gts = categ + '/GT'

    for name in sorted(os.listdir(raindrop)):
        if not name.startswith('.'): #Avoid .DS_STORE

            rain = os.path.join(raindrop, name)
            gt = os.path.join(gts, name)


            rain_img = cv2.imread(rain)
            gt_img = cv2.imread(gt)

            #Computing image difference
            de_gray = cv2.cvtColor(rain_img, cv2.COLOR_BGR2GRAY)
            gt_gray = cv2.cvtColor(gt_img, cv2.COLOR_BGR2GRAY)
            diff = (de_gray-gt_gray)

            diff = cv2.cvtColor(diff,cv2.COLOR_GRAY2RGB)
            diff = cv2.applyColorMap(diff, cv2.COLORMAP_JET)
            horizontal = np.hstack((rain_img, gt_img,diff))

            while True:
                cv2.imshow('img', horizontal)

                k = cv2.waitKey(0)
                if k== ord('q'):
                    break
                elif k == ord('d'):
                    os.remove(degrade_file)
                    os.remove(gt_file)
                    break
                elif k == ord('s'):
                    saveImg(degrade_img,gt_img)
                    break
                elif k == ord('k'): #Exit on k
                    sys.exit()
```

Bibliography

[1] Juan F P J Abascal, S Bussod, N Ducros, S Si-Mohamed, P Douek, et al. A residual u-net network with image prior for 3d image denoising. In *28th European Signal Processing Conference (EUSIPCO)*, Amsterdam, Netherlands, 2021.

[2] Stefano Alletto, Casey Carlin, Luca Rigazio, Yasunori Ishii, and Sotaro Tsukizawa. Adherent raindrop removal with self-supervised attention maps and spatio-temporal generative adversarial networks. pages 2329–2338, 10 2019.

[3] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[6] Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. Dehazenet: An end-to-end system for single image haze removal. *CoRR*, abs/1601.07661, 2016.

[7] Zhixiang Chi, Yang Wang, Yuanhao Yu, and Jin Tang. Test-time fast adaptation for dynamic scene deblurring via meta-auxiliary learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9137–9146, 2021.

[8] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. *CoRR*, abs/2108.05054, 2021.

[9] Tali Dekel, Dilip Krishnan, Hossein Mobahi, Tucker Freeman, and William T Freeman. Learning to see through obstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12256–12265, 2020.

[10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[11] David Eigen, Dilip Krishnan, and Rob Fergus. Restoring an image taken through a window covered with dirt or rain. In *2013 IEEE International Conference on Computer Vision*, pages 633–640, 2013.

[12] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[14] Jochen Fröhlich. Neural net components in an object oriented class structure. Neural Networks with Java, 1996.

[15] K. Garg and S.K. Nayar. Detection and removal of rain from videos. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004.

[16] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 2008.

[17] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press,

2016.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[19] Xiefan Guo, Hongyu Yang, and Di Huang. Image inpainting via conditional texture and structure dual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14134–14143, 2021.

[20] Da He, Xiaoyu Shang, and Jiajia Luo. Adherent mist and raindrop removal from a single image using attentive convolutional network. *Neurocomputing*, 505:178–187, 2022.

[21] Da He, Xiaoyu Shang, and Jiajia Luo. Adherent mist and raindrop removal from a single image using attentive convolutional network. *Neurocomput.*, 505(C):178–187, sep 2022.

[22] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1956–1963, 2009.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[25] Matheus Jacques. Batch vs mini-batch vs stochastic gradient descent with code examples. medium, 2022. `https://medium.datadriveninvestor.com/batch-vs-mini-batch-vs-stochastic-gradient-descent-with-code-examples-cd8232174e14`, Accessed: 2023-04-05.

[26] Long Jing and Yingli Tian. Self-supervised learning: Generative or contrastive. *arXiv*

preprint arXiv:2006.08218, 2020.

[27] Eun Seo Jo and Timnit Gebru. Lessons from archives: Strategies for collecting sociocultural data in machine learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 306–316, New York, NY, USA, 2020. Association for Computing Machinery.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[30] John R. Koza, Forrest H. Bennett, David Andre, and Martin A. Keane. *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*, pages 151–170. Springer Netherlands, Dordrecht, 1996.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[32] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8878–8887, 2019.

[33] Amjad J. Humaidi Ayad Al-Dujaili Ye Duan Omran Al-Shamma J. Santamaría-Mohammed A. Fadhel Muthana Al-Amidie Laith Alzubaidi, Jinglan Zhang and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8:53, 2021.

[34] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[35] Jingyuan Li, Chenxi Liu, and Jian Peng. Meta-auxiliary learning for one-shot semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14697–14707, 2021.

[36] Qiang Li, Yuanming Cao, Zhixiang Chi, and Xiaolin Wu. Seeing through unclear glass: Occlusion removal with one shot. unpublished, 2023.

[37] Hongyu Liu, Bin Jiang, Yi Xiao, and Chao Yang. Coherent semantic attention for image inpainting. *CoRR*, abs/1905.12384, 2019.

[38] Shikun Liu, Andrew J. Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. *CoRR*, abs/1901.08933, 2019.

[39] Wei Lu. Neural network model for distortion buckling behaviour of cold-formed steel compression members. Technical Report 16, Helsinki University of Technology Laboratory of Steel Structures, 2000.

[40] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[41] Anthony J. papagelis and Dong Soo Kim. Artificial neural networks, 2021. `https://www.cse.unsw.edu.au/~cs9417ml/MLP2/`, Accessed: 2023-04-05.

[42] Anthony J. papagelis and Dong Soo Kim. Backpropagation in multilayer perceptrons, 2021. `https://www.cse.unsw.edu.au/~cs9417ml/MLP2/BackPropagation.html`, Accessed: 2023-04-05.

[43] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016.

[44] Rui Qian, Robby T. Tan, Wenhan Yang, Jiajun Su, and Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. *CoRR*, abs/1711.10098, 2017.

[45] Yuhui Quan, Shijie Deng, Yixin Chen, and Hui Ji. Deep learning for seeing through

window with raindrops. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[46] Dongwei Ren, Wangmeng Zuo, Qinghua Hu, Pengfei Zhu, and Deyu Meng. Progressive image deraining networks: A better and simpler baseline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3937–3946, 2019.

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

[48] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016.

[49] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[50] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[51] U. Sara, M. Akter, and M.S. Uddin. Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 7:8–18, 2019.

[52] Irhum Shafkat. Intuitively understanding convolutions for deep learning, 2018. `https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1`, Accessed: 2023-04-05.

[53] Wenqi Ren Zhangyang Wang-Eric K. Tokuda Roberto Hirata Junior Roberto Cesar-Junior Jiawan Zhang Xiaojie Guo Siyuan Li, Iago Breno Araujo and Xiaochun Cao. Single image deraining: A comprehensive benchmark analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[54] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[55] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9229–9248. PMLR, 13–18 Jul 2020.

[56] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *International Conference on Robotics and Automation (ICRA 2018)*. IEEE, 2018.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[58] Solegaonkar Vikas K. Introduction to pytorch, 2019. `https://towardsdatascience.com/introduction-to-py-torch-13189fb30cb3`,Accessed: 2023-04-05.

[59] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[60] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, 2:1398–1402, 2003.

[61] Wenhui Wei, Wenhan Yang, and Jiaying Liu. Deep joint rain detection and removal

from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1357–1366, 2017.

[62] Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io*, 2018.

[63] Badr Will. Auto-encoder: What is it and what is it used for? (part 1), 2019. `https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726`,Accessed:2023-04-06.

[64] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30:79–82, 2005.

[65] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[66] Yujie Wu, Hong Zhang, Yawei Li, Yinan Mao, Lei He, and Zhoufeng Liu. Dense block u-net for dynamic scene deblurring. In *Proceedings of the 2020 The 4th International Conference on Graphics and Signal Processing*, pages 46–50, 2020.

[67] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[68] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[69] Shaodi You, Robby T. Tan, Rei Kawakami, and Katsushi Ikeuchi. Adherent raindrop modeling, detection and removal in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1721–1733, 2016.

[70] He Zhang and Vishal M. Patel. Densely connected pyramid dehazing network. *CoRR*,

abs/1803.08396, 2018.

[71] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, jul 2017.

[72] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.