UNIVERSITY of PADOVA

DEPARTMENT of INFORMATION ENGINEERING

MASTER THESIS IN ICT FOR INTERNET AND MULTIMEDIA

LEARNING-BASED LOW LIGHT IMAGE DENOISING

SUPERVISOR

PROF. PIETRO ZANUTTIGH

UNIVERSITY of PADOVA

MASTER CANDIDATE

ESRAA M. B. ALNAJJAR

CO-SUPERVISOR

MAZEN MEL

PHD STUDENT

STUDENT ID

2009462

ACADEMIC YEAR

2022-2023

Acknowledgment

First of all, I would like to praise and thank Almighty God, who has given me countless blessings, knowledge, and opportunity, so that I can finally complete the thesis.

Words cannot express my gratitude to my mentor, Pietro Zanuttigh, for his patience and invaluable feedback. I also could not have made this trip without the guidance of Mr. Mazen Mel, who generously provided knowledge and experience.

In addition, this endeavor would not have been possible without the support of the University of Padua faculty, for giving me the opportunity to do my research and providing all the resources. Thanks also to all the Professors who influenced and inspired me.

I am also grateful to my family and friends for believing in me and keeping my spirits high and excited throughout the process. Without their encouragement and motivation, I would not have been able to complete this journey.

Abstract

The need for high quality imaging devices beyond the limitations imposed by the hardware and optics of such systems is essential for a variety of tasks such as in the medical field, monitoring, and industrial applications. In some cases, such images are captured under low-light conditions resulting in low contrast, low brightness and significant amount of noise, which leads to extremely poor image quality and difficulty in extracting meaningful information from such data.

In this work, a deep learning-based low light image denoising and enhancement method is investigated, which can reduce noise and improve the image contrast at the same time. The method contains two deep sub-networks, an Image Denoising Network and an Enhancement Network. The Denoising network takes a noisy low-light image as input and produces a denoised one while the Enhancement network takes the resulting denoised image as input and produces a brighter enhanced one.

List of Figures

Figure 1.1 Image formation …… **3**

Figure 3.1 Lol dataset samples ……..……. ***9***

Figure 3.2 Bayer pattern .....……..……... **10**

Figure 3.3 Summary of normalization techniques …….…… **12**

Figure 3.4 Data processing pipeline …….. **14**

Figure 4.1 Schematic diagram of backpropagation training algorithm…… **15**

Figure 4.2 Residual learning……...……. **16**

Figure 4.3 Residual block …….. **17**

Figure 4.4 Unet architecture ……..... **18**

Figure 4.5 ResUnet architecture ……...……. **20**

Figure 5.1 Flowchart of low-light image denoising and enhancement……. **22**

Figure 5.2 Architecture of the proposed denoising model ……..……. **25**

Figure 5.3 Loss curves……... **30**

Figure 5.4 PSNR and SSIM curves …… **31**

Figure 5.5 ResUnet Loss Curve……..… **31**

Figure 5.6 Test samples and denoised outputs ……... **32**

Figure 5.7 Noisy and denoised samples..... **33**

Figure 6.1 Histogram of an image before and after equalization ……. **34**

Figure 6.2 Histogram equalization enhanced results ……..... **36**

Figure 6.3 Framework of zero-reference deep curve estimation …….…..... **38**

Figure 6.4 Deep curve estimation network architecture …….……..…. **40**

Figure 6.5 Zero-dce loss curves …… **45**

Figure 6.6 Zero-dce enhanced results ……..……..… **47**

Figure 6.7 MIRNet architecture…….…... **48**

Figure 6.8 Selective kernel feature fusion..... **50**

Figure 6.9 Dual attention unit ……..... **51**

Figure 6.10 Multi-Scale residual block..... **51**

Figure 6.11 MIRNet loss curve……..... **53**

Figure 6.12 MIRNet enhanced results……..... **54**

Figure 6.13 Final ResUnet output ……..……. **55**

Figure 6.14 Visual comparison with enhancement methods..... **56**

List of abbreviations

3-D	Three Dimension	**GPU**	Graphics processing unit
2-D	Two Dimension	**MAE**	Mean absolute error
CCD	Charge-coupled device	**Vgg**	Visual geometry group
CMOS	Complementary metal–oxide–semiconductor	**MIRNet**	Multi-scale information retention network
RGB	Red Green Blue	**Zero-dce**	Zero-reference deep curve estimation
CFA	Color filter array	**HE**	Histogram equalization
ISO	sensitivity to light Tomography	**CDF**	Cumulative distibution function
AWGN	Additive white gaussian noise	**HSV**	Hue,Saturation and value color space
LOL	Low-light dataset	**LEC**	Light enhancement curve
ReLU	Rectified linear unit	**CNN**	Convolutional neural network
ResNets	Residual networks	**DCNN**	Deep Convolutional Neural Network
ResUNet	Residual U-Net	**RRGs**	Recursive residual groups
BN	Batch Normalization	**SKFF**	Selective Kernel Feature Fusion
PSNR	Peak signal to noise ratio	**GAP**	Global Average Pooling
SSIM	Structural similarity	**DAU**	Dual attention unit
MSE	Mean square error	**MRB**	Multi-scale residual block

List of Tables

Table 5.1	Convolutional layers in residual units of denoising model…..……..	**24**
Table 5.2	Comparison between different Loss functions…..……..…..……..	**28**
Table 6.1	Zero-DCE layers.....	**41**
Table 6.2	Evaluation metrics of ResUnet ……..…....	**55**
Table 6.3	Quantitative comparison on LOL dataset in terms of PSNR, SSIM ...	**56**

Contents

Acknowledgmet ..... **II**

Abstract ..... **II**

List of Figures ..... **III**

List of abbreviations ..... **V**

List of Tables ..... **VI**

Contents ..... **VII**

Introduction **1**

1.1 Digital Cameras 1

1.1.1 Digital image formation pipeline 2

1.1.2 Limitations of digital image sensors 4

1.1.3 Noise modeling 4

1.2 Thesis overview 5

Related Work **6**

2.1 Model-based image denoising and enhancement 6

2.2 Deep learning for low-light Image Denoising and enhancement 7

Dataset **10**

3.1 Overview 10

3.2 Raw Images 11

3.3 Data Preprocessing 12

3.3.1 Data Generator 12

3.3.2 Data Normalization 12

3.3.3 Data Augmentation 13

3.4 Shot and Read Noise Model 14

Residual Learning **16**

4.1 Residual Learning in Deep Learning 16

4.2 Residual Learning Formulation 18

4.3 Networks Architecture 18

4.3.1 UNet Architecture 19

4.3.2 ResUNet Architecture 21

Proposed Denoising Model **23**

5.1 Methodology 23

5.1.1 ResUnet Model 23

5.1.2 Loss Functions 27
5.1.3 Evaluation Metrics 27
5.2 Experimental setup 29
5.2.1 Dataset 29
5.2.2 Implementation details 29
5.3 Denoising Results 29
5.3.1 Sample Results 33
Enhancement Techniques 35
6.1 Histogram Equalization 35
6.1.1 Histogram Equalization Result 37
6.2 Zero-Reference Deep Curve Estimation 38
6.2.1 Zero-DCE Framework 39
6.2.2 Zero-DCE Experiment and Results 45
6.3 Multi-Scale Information Retention Network 48
6.3.1 MIRNet Framework 49
6.3.2 MIRNet Experiment and Results 53
6.4 Results Comparison and Evaluation 56
6.4.1 Denoising phase: 56
6.4.2 Enhancement phase: 57
Conclusion 59
Bibliography 60

1 Introduction

In recent years, the widespread use of digital cameras and smartphones has made photography an integral part of everyday life. However, capturing images in low light conditions remains a challenge due to the limited amount of available light. This can lead to images that are noisy, blurred, and lack details, making it difficult to perceive the scene and appreciate its visual qualities.

Low-light image denoising and enhancement are two important tasks in image processing that aim to improve the quality of images captured in low-light conditions. Various approaches have been proposed to tackle these tasks, ranging from traditional signal processing methods to more recent machine learning-based techniques.

In this thesis, we focus on learning-based methods for low-light image denoising and enhancement. Specifically, we explore the use of deep neural networks, which have shown remarkable performance in various computer vision tasks. We investigate architectures and loss functions that are tailored to the low-light image enhancement and demonstrate their effectiveness through extensive experiments on a benchmark dataset.

The remainder of this thesis is organized as follows: we first review related work in low-light image processing and deep learning. Then, we describe our proposed methods for low-light image denoising and enhancement, followed by a thorough evaluation of their performance. Finally, we discuss the limitations of our approach and suggest possible directions for future research.

1.1 Digital Cameras

The widespread use of multimedia applications today, has led to the usage of digital camera as an essential component in various portable devices such as smartphones, webcams and gaming devices. A digital camera is an integrated system constituting of a lens, sensor, and digital image processor, each of which is a sophisticated system on its own. A digital camera system is a hardware device that designed based on principles of optics and device physics, a camera takes the trace of light from a scene and passes it through the lens to the sensor. In modeling any image formation pipeline, geometric primitives and transformations are crucial to project 3-D geometric features into 2-D features. However, apart from geometric features, image formation also depends on discrete color and intensity values.

1.1.1 Digital image formation pipeline

In imaging system, Fig. 1.1 (a) gives a simple explanation of image formation, where the light from a physical source falls on the object, part of this light reflects on a particular surface after striking the object and goes through an image plane that reach a sensor plane via the lens (optics).

From a viewpoint of color, we know visible light is only a small portion of a large electromagnetic spectrum, so Colour sensors are used to recognize a material's color in the RGB (red, green, blue) spectrum, while rejecting unwanted infrared or ultraviolet light.

Bayer Grid/Filter is an important development to capture the color of the light. In a camera, RGB sensors capture all three-color primaries. Inspired by human visual preceptors, Bayer proposed a grid in which there are 50% green, 25 % red, and 25% blue color filters mounted on top of the sensor.

A **demosaicing algorithm** is then used to obtain a full-color image where the surrounding pixels are used to estimate the values for a particular pixel.

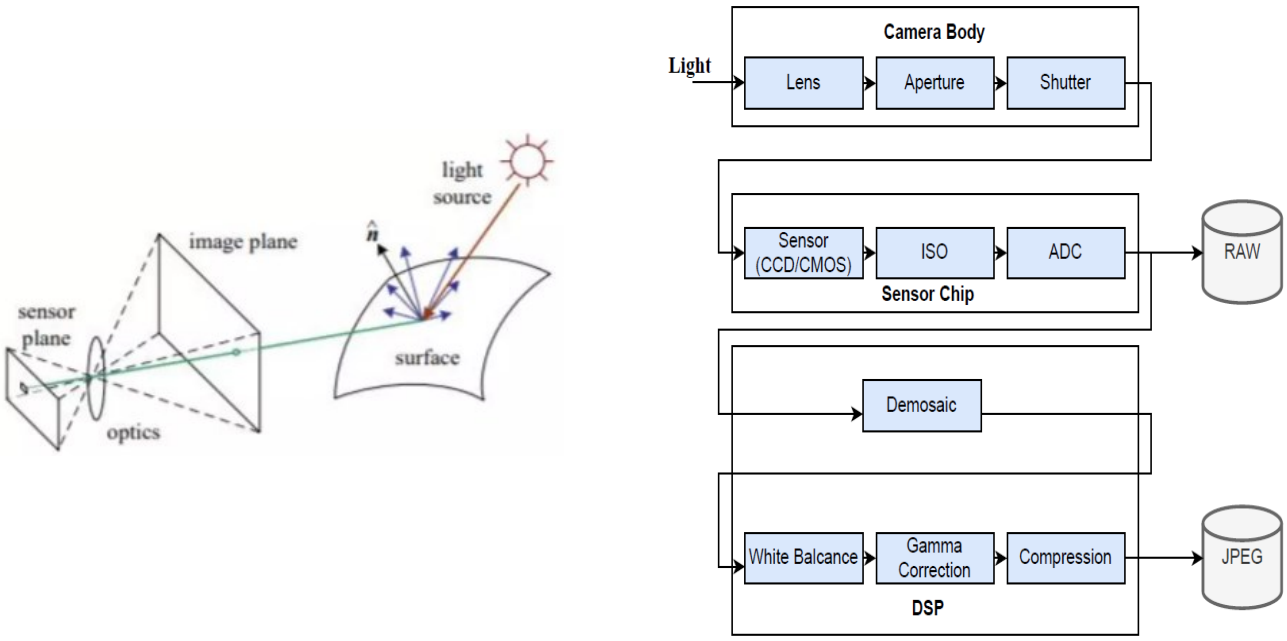
There are many other color filters that have been developed to sense colors apart from Bayer Filter.

In digital camera, the light arises from multiple light sources, reflects on multiple surfaces, and finally enters the camera where the photons are converted into the (R, G, B) values that we see while looking at a digital image. Fig. 1.1(b) shows the image sensing pipeline where the light first falls on the lens. Following the aperture and shutter that can be adjusted. Then the light reaches an image sensor which can be CCD or CMOS sensor. This sensor is actually in the shape of array or a rectangular grid, where each cell in the array is light-sensitive diode that senses the intensity of photon and converts it to electrons. Then the image is obtained in an analog or digital form, and we get the raw image.

In the case of CCD, a charge is generated at each sensing element and this photogenerated charge is moved from pixel to pixel and is converted into a voltage at the output node. Then an analog to digital converter (ADC) converts the value of each pixel to a digital value while in CMOS sensors a charge converted to voltage inside each element so we get a digital signal. Both CCD and CMOS are gray-scale sensors, senses the number of photons not their wavelength, and to get the colored image they use a color filter array

(CFA) which is a mosaic of tiny color filters placed over the image sensor to capture color information.

Next, white balancing and other digital signal processing operations are done and the image is finally compressed to a suitable format and stored.



(a) (b)

Figure 1.1: (a) Image formation (b) Image sensing pipeline in a camera [48]

1.1.2 Limitations of digital image sensors

During image acquisition and transmission, noise is introduced into the image. The introduction of noise into the image could be caused by several factors such as: low-light conditions, sensor temperature, dust particles, transmission channel interference and other environmental factors that may have an impact on the imaging sensor.

Digital images acquired by the sensor need to be processed before obtaining the final image. And due to the random absorption of photons, the image capture is always subject to noise which can be amplified by the processing operations.

Noise in image sensor is typically separated into two categories: random noise and fixed pattern noise.

Random noise produces statistically random variations in the brightness and color of an image both above and below the actual image intensity. Although the pattern of random noise varies between photos shot with identical exposure settings, the magnitude of the noise will remain the same. Random noise is present in all digital images and is especially prevalent in high ISO and short-duration exposures. It's primarily caused by shot noise and read noise. While the **Fixed-pattern noise** is a term given to a particular noise pattern on digital imaging sensors often noticeable during longer exposure shots where particular pixels are susceptible to giving brighter intensities above the average intensity. Although fixed pattern noise appears more displeasing, it is usually easier to remove since it is repeatable. A camera's internal electronics just has to know the pattern and it can subtract this noise away to reveal the true image. Fixed pattern noise is much less of a problem than random noise in the latest generation of digital cameras.

However, it is usually more difficult to remove random noise without affecting image quality. Software algorithms find it difficult to distinguish between random noise and the texture pattern, so removing this noise may end up removing the texture as well.

1.1.3 Noise modeling

In computer vision, noise modeling refers to the process of capturing and incorporating the random and often unpredictable distortions or corruptions in an image or video data that can occur during acquisition, transmission, or processing. These distortions can be caused by various factors such as sensor noise, environmental factors, compression, or hardware limitations.

Noise modeling is important in computer vision because it enables researchers and engineers to simulate and analyze the impact of noise on image or video data, and to

develop algorithms and techniques that can effectively mitigate the effects of noise. This is especially important in applications such as medical imaging, surveillance, or autonomous driving, where the accuracy and reliability of computer vision systems can be critical.

Noise models can vary depending on the application and the specific type of noise being considered. Common noise models used in computer vision include additive white Gaussian noise (AWGN), Rayleigh Noise, Erlang (or gamma) Noise, Exponential Noise, Uniform Noise, and Impulse Noise, among others. By incorporating these noise models into their algorithms and testing their performance under different noise conditions, researchers and engineers can better understand the limitations of their systems and optimize them for real-world application.

1.2 Thesis overview

Denoising low-light images is a challenging task arising from the various sources of noise. In low light conditions, digital cameras increase the ISO (electronic gain) to amplify the brightness of captured images. However, this in turn amplifies the noise, arising from read and shot sources. In the raw domain, read and shot noise are effectively modeled using Gaussian and Poisson distributions respectively.

In low-light imaging, noise removal becomes a critical challenge to produce a high-quality, detailed image with low noise. In this work, we use a residual neural network that combines the strengths of residual learning and U-Net [49] for image denoising. The network is built with residual units and has similar architecture to that of U-Net. The benefits of this model have two parts: first, residual units ease the training of deep networks. Second, the rich skip connections within the network could facilitate information propagation, allowing us to design networks with fewer parameters and better performance.

For low light enhancement, we will use multiple techniques, starting from traditional methods like histogram equalization and moving toward state-of-the-art deep learning approaches.

2 Related Work

Image Denoising is the process of recovering a clean image from its noisy observation. It is a crucial research topic in the fields of computer vision and image processing due to its importance in many high-level applications such as image encryption, bioinformatics, texture classification and many others.

In recent decades, we have witnessed great progress in image denoising and low light enhancement techniques models starting from traditional model-based methods toward deep learning-based models which offer fast inference and good performance.

2.1 Model-based image denoising and enhancement

Various techniques of noise reduction have been developed from different points of view, such as image filtering, shrinkage of coefficients in transform domains, sparse representation of a learned dictionary, and non-local self-similarity statistics. Representation methods include bilateral filtering [1], non-local means (NLM) [2], markov random field (MRF) models, block matching and 3D filtering (BM3D) [3], K-SVD [4],higher-order singular value decomposition (HOSVD) [5], and weighted nuclear norm minimization (WNNM) [6].To improve the performance of the approaches mentioned above, Schmidt, Roth [7] proposed a cascade of shrinkage fields (CSF) for image denoising, a kind of unified random field model. Recently, Chen and rock [8] developed a trainable nonlinear reaction-diffusion (TNRD) model. These classical methods performed well in denoising however, they may require manual parameter setting and are computationally expensive.

As for the traditional model-based methods which are suitable for low-light image enhancement, several methods have been proposed. Gamma Correction used to correct the differences between the way a camera captures content, the way a display displays content, and the way our visual system processes light, Rahman et al. [9] proposed an adaptive gamma correction method that dynamically determines the intensity conversion function based on the statistical characteristics of the image.

Histogram Equalization [10] rearrange the pixel values of low light image to improve the contrast and brightness of the image. Adaptive Histogram Equalization (AHE)

[11] has been proposed to enhance the contrast locally. However, AHE overamplifies the noise in relatively homogeneous region of the image and to prevent this a Contrast Limited Adaptive Histogram Equalization (CLAHE) [12] is used.

Retinex-based methods [13][14] decompose the low light image into reflectance and illumination maps and obtain the enhanced image by fusing the restored reflectance map and the illumination maps.

Dehazing-based methods [15][16] improve the visibility of the image by considering the inverted low-light image as a haze and applying the dehazing.

Although these methods can improve the brightness for the dark pixels, they barely regard the realistic lighting conditions and produce enhanced images that are inconsistent with the actual scene.

2.2 Deep learning for low-light Image Denoising and enhancement

Deep learning technologies is widely used in various fields, achieving promising results in image denoising and enhancement. Since the proposal of big data analysis and Graphic Processing Unit (GPU), deep learning techniques received a great deal of attention and has been widely applied in the field of image processing.

In the fields of image processing and computer vision, Image Denoising has been a hot topic for a long time, therefore, numerous deep learning-based models have achieved remarkable success in this field. In [35], Jain and Seung use convolutional neural networks (CNNs) for image denoising and claimed that CNNs have similar or even better representation power than the MRF [36] model. A deep convolutional neural network for image denoising (DnCNN) was proposed by Zhang et al. [17], this model employs bundle of convolutional layers, rectified linear unit activations (ReLU), batch normalization [18] and the use of residual learning to improve the performance of denoising. In [19] Tai et al. proposed a deep end-to-end persistent memory network for image denoising, this model fuses both short-term and long-term memories to capture different levels of information. The fast and flexible denoising CNN (FFDNet) proposed by Zhang et al. [20] introduces noise feature map for handling non-uniform noise level and downsampled sub-images for increasing the receptive field and the performance speed. The work in [21] proposes a deep residual CNN (DRCNN) based on DnCNN and ResNet [22] architecture to get rid of gradient vanishing problem caused by increasing the network depth. Despite their excellent performance, these models' complexity and number of parameters were increased to achieve such an improvement.

Additionally, to boost the performance, several recent methods apply techniques such as residual learning and skip connections. The Unet model [23] is one of widely used

autoencoder architectures today, the encoder block uses several convolutions and max-pooling that halve the size of feature maps and double the number of these maps while the decoder restores the size of feature maps and reuse it by the use of skip connection leading to a reduction in the loss of information that is caused by the encoding process.

Deep learning-based Methods have a brilliant impact on low-light image enhancement. Lore et al. [24] proposed the first convolutional neural networks for low-light image enhancement termed (LL-Net) that performs contrast enhancement and denoising based on deep auto-encoder. Chen et al. [25] proposed Retinex-Net, which mainly has two networks, Decom-Net decomposes the input images into reflectance and illumination maps, and Enhance-Net that adjusts the illumination map to get the final enhanced image. Guosheng Lin et al. in [37] proposed a refined network LL-RefineNet which is built to learn from the synthetical dark and noisy training images, and perform image enhancement for natural low-light images in symmetric-forward and backward-pathways.

In KinD [28] Zhang et al. inspired by the Retinex-Net and presented a new decomposition network, a reflection enhancement network and an illumination map enhancement network, which achieved outstanding performance in low light image enhancement. Wang et al. proposed an end-to-end enhancement network [39], which is composed of Retinex decomposition network (RDNet) and fusion enhancement network (FENet). After inputting the image, it is decomposed into illumination component and reflection component by RDNet, and then the decomposed illumination component is preliminarily enhanced by the camera response function. Finally, the input image, the decomposed reflection component, and the preliminary enhanced illumination component are used as the input to FENet for fusion enhancement, and the final enhancement result is obtained. Zhu et al. [40] proposed the low-light enhancement method of EEMEFN. The network framework is divided into multi exposure fusion (MEF) which generates multiple exposure images through a given exposure ratio, and edge enhancement (EE) that fuses the different scale information of the generated multiple exposure images through the U-net structure.

In [26] Guo et al. proposed a zero-shot learning method called Zero-DCE which heavily relies on the usage of multi-exposure training data. The work in [27] proposes a backbone model EnlightenGAN which relies on large number of parameters for good performance. Chen Wei et al [41] proposed RRDNet that uses a recurrent residual dense block to process the input low-light image. The block consists of multiple dense convolutional layers and residual connections to allow information flow between layers, while also avoiding the vanishing gradient problem.

In [42] Zamir et al. proposed a multi-scale information retention which (MIRNet) introduces a new design that aims to preserve high-resolution features across the entire network, while also leveraging rich contextual information from the low-resolution representations. The dual illumination estimation network (DIEN) [43] aims to enhance low-light images by using a dual-pathway network to extract features from the input image

under both illumination conditions. These features are then combined and used to generate the enhanced image.

3 Dataset

3.1 Overview

Low light datasets are often used in computer vision research, particularly to evaluate the performance of different algorithms and techniques for image enhancement, denoising, and object detection in low light conditions. These datasets can be captured in a variety of low light conditions or by simulating low light conditions through post-processing techniques.

In this work, we will evaluate the proposed models using the **LOL** (**LO**w-**L**ight) [29] dataset which contains 500 low-light and normal-light image pairs of different kinds such as toys, books, garden, etc. It is divided into 485 training pairs and 15 testing pairs. All the images have a resolution of 400×600. Fig. 3.1 shows some samples.

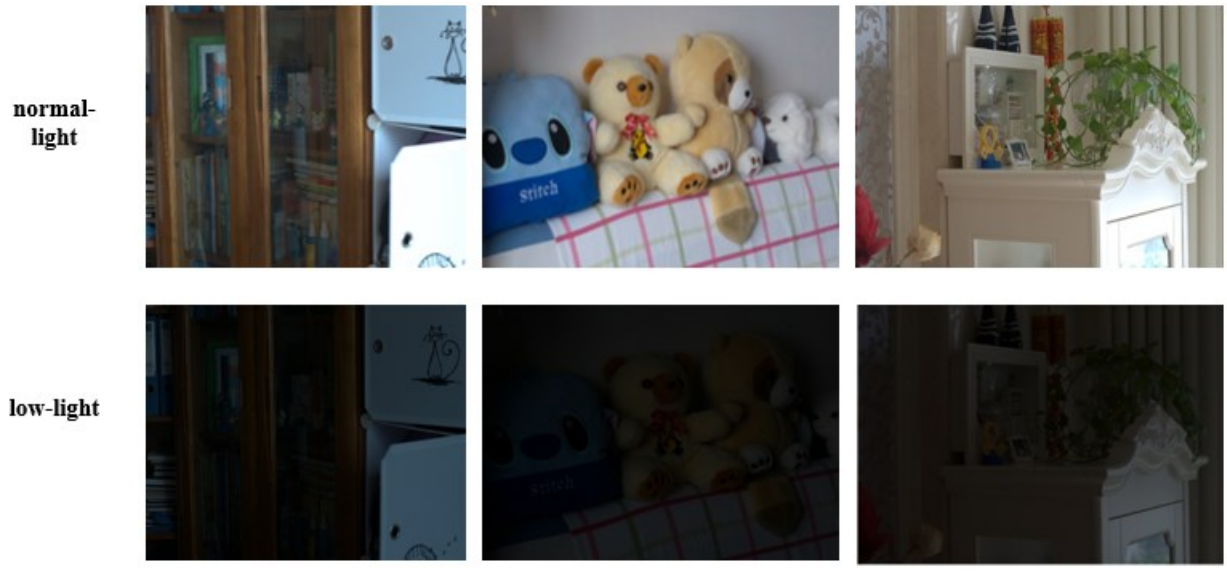


Figure 3.1: LOL dataset samples

3.2 Raw Images

RAW images are a digital camera-specific format that typically contains unprocessed image containing both the sensor pixel values and all meta-information about the image generated by the camera. Raw files come in many proprietary file formats (Nikon's. NEF, Canon's .CR2, etc.) these files constitute a repository of all the captured information of the scene.

The image sensor's Raw data contains the light intensity values captured from the scene, this data is a single channel intensity image, and typically comes in the form of color filter array (CFA) which is an $m * n$ array of pixels (where m and n are sensor's dimensions), each pixel carries information about a single-color channel (Red, Green or Blue). Since the light falls on any given pixel in CCD sensor is recorded as a number of electrons in a capacitor, it can be saved only as a scalar value; and hence a single pixel can't reflect the three-dimensional nature of the light. CFAs offer a compromise where information about each of the three-color channels is captured at different locations by means of spectrum-selective filters placed over each pixel. Bayer array (shown in Fig. 3.2) is the most common CFA pattern used in the modern digital cameras, it has twice as many pixels represent green light because the human eye is more sensitive to contrast in shades of green and closely correlates with the light intensity perception of a scene.

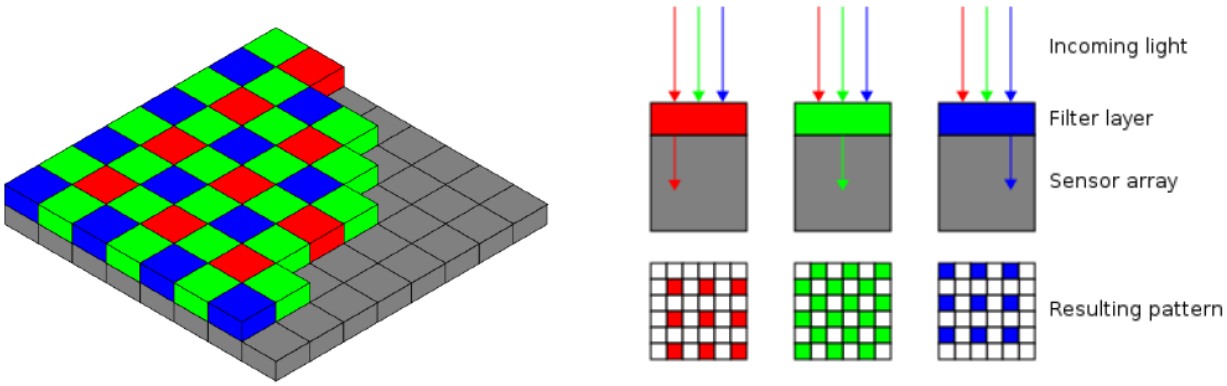


Figure 3.2: Bayer pattern [50]

3.3 Data Preprocessing

Data pre-processing is the first crucial step in converting raw data into processed data that can be fed into the model.

3.3.1 Data Generator

Using a data generator to load data is imperative while training deep learning models as we need large dataset to train good models, so there is a need for large memory requirements to analyze and process this extremely large amount of data. But due to the limitations of our memory, it is not possible to load the whole data into memory; therefore, we used a data generator to get around this issue.

The data generator is a function that generates batches of input data and its corresponding target data for training a neural network. It loads a batch of data into memory and feeds it to the model for training, then discards it and loads the next batch. This allows the model to be trained on large dataset without running out of memory.

3.3.2 Data Normalization

In deep learning, data normalization refers to the process of transforming input data to have zero mean and unit variance. It is one of the most important stages of data preparation for training neural networks, it can improve the performance and the stability of the model during training.

Normalization assigns equal weights/importance to each variable, ensuring that no single variable steers model performance in one way simply because it is larger. There are four typical normalization techniques that may be useful:

- **Min-Max Normalization**: is one of the most popular methods for normalizing data for each feature, the lowest value is converted to a 0, the maximum value is converted to a 1, and all other values are converted to a decimal between 0 and 1. It is formulating as:

$$x' = (x - x_{min})/(x_{max} - x_{min}) \quad (1)$$

- **Z-Score Normalization**: This technique, also known as standardization. It's common in machine learning methods like SVM and logistic regression and expressed as follows:

$$x' = (x - \mu)/\sigma \quad (2)$$

Here, x' represents the standard score, μ the population mean, and σ the population standard variation.

- **Log Scaling**: is a method that uses logarithms to compress a wide range into a smaller range. As a result, the distances between the data before and after scaling may not be proportionate. It is ideal for detecting many natural occurrences, it formulated as:

$$x' = log(x) \quad (3)$$

- **Feature Clipping**: is the process of removing data points beyond a certain minimum or maximum. It's useful for removing extreme outliers from a data set. It is typically done during the data preprocessing step.

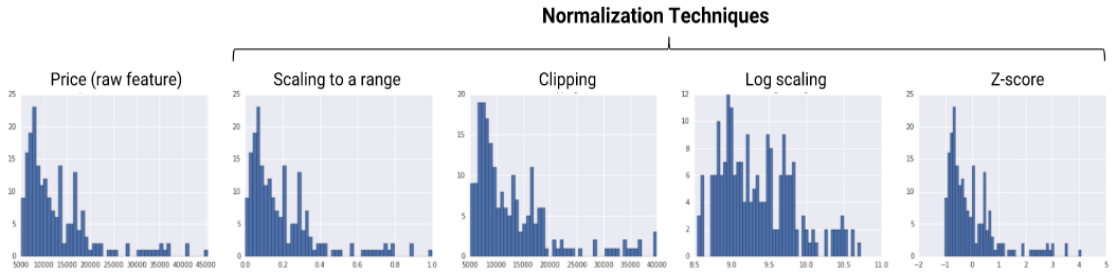


Figure 3.3: Summary of normalization techniques [51]

3.3.3 Data Augmentation

Data augmentation is a technique generally used in deep learning to increase the size of a dataset by applying different transformations to the existing dataset.

The main idea behind data augmentation is to produce a model that can be more robust and generalize better to a new dataset, by introducing slight variations to the existing dataset.

Data augmentation is most appropriate when the existing dataset is small, as it allows the model to learn from a more diverse and representative set of examples. It is often used in computer vision tasks such as image classification, object detection, and natural language processing tasks, where new data is created by some transformations applied on the input dataset, some of common transformations used in computer vision tasks include:

- Random **cropping**: randomly cropping a region of the image

- Random **flipping**: randomly flipping the image horizontally or vertically
- Random **rotation**: randomly rotating the image by a small angle
- Random **scaling**: randomly scaling the image by a small factor

By combining these transformations, a large number of new training examples can be created from the original dataset, which can improve the performance of the model and reduce the risk of overfitting.

Moreover, we can apply some other transformation like **changing the brightness or contrast** of the original data, and also in some cases data augmentation can involve **adding noise** to the original dataset.

3.4 Shot and Read Noise Model

The shot noise and read noise model is a widely adopted model in the field of signal processing and imaging that describes the sources of the actual noise in an image.

Shot noise, also referred to as photon noise, originates from the random nature of light when it is exposed by a camera sensor. This type of noise is proportional to the square root of the number of photons detected by the sensor, and it is generally the dominant source of noise in bright image regions.

Read noise, on the other hand, is a type of electronic noise that arises during the process of reading out the signal from the camera sensor. It is a constant noise level that is present in all images captured by the sensor, and it is typically more significant than shot noise in well-light imaging situations.

In this work, inspired by the noise model proposed in [31] we simulate the noise in raw sensor, where shot noise represents a Poisson random variable whose mean is the true light intensity, whereas read noise is a Gaussian random variable with zero mean and fixed variance. Together shot and read noise can be modeled as a single heteroscedastic Gaussian, where the noise values are independent and identically distributed random variables with zero mean and a certain standard deviation.

The observed intensity $\boldsymbol{y}$ is treated as a random variable whose variance is a function of the input intensity $\boldsymbol{x}$,

$$\boldsymbol{y} \sim \mathcal{N}(\mu = \boldsymbol{x}, \boldsymbol{\sigma^2} = \boldsymbol{\lambda_{read}} + \boldsymbol{\lambda_{shot}}\,\boldsymbol{x}) \qquad (4)$$

Where λ_{read}, λ_{shot} are determined by the sensor's analog and digital gains. For digital gain g_d, analog gain g_a and certain sensor readout variance σ_r^2, we have:

$$\boldsymbol{\lambda_{read} = g_d^2 \sigma_r^2, \quad \lambda_{shot} = g_d g_a} \tag{5}$$

To choose noise levels for our input images, we generate random noise levels from a log-log linear distribution, then model the joint distribution of different read/shot noise parameter pairs in the real raw images of the LOL dataset.

$$\begin{aligned} \log(\lambda_{shot}) &\sim \mathcal{U}(a = \log(0.0001), b = \log(0.012)) \\ \log(\lambda_{read}) \mid log(\lambda_{shot}) &\sim \mathcal{N}(\mu = 2.18 \log(\lambda_{shot}) + 1.2, \sigma = 0.26) \end{aligned} \tag{6}$$

In short, we can summarize the data processing stage as shown in Fig.4, where we applied data augmentation on the original LOL dataset, and then used the noise model to generate random noise that simulates the noise generated by the camera sensor, and finally, we used the data generator to feed the denoising model with a pair of the clean and noisy image.

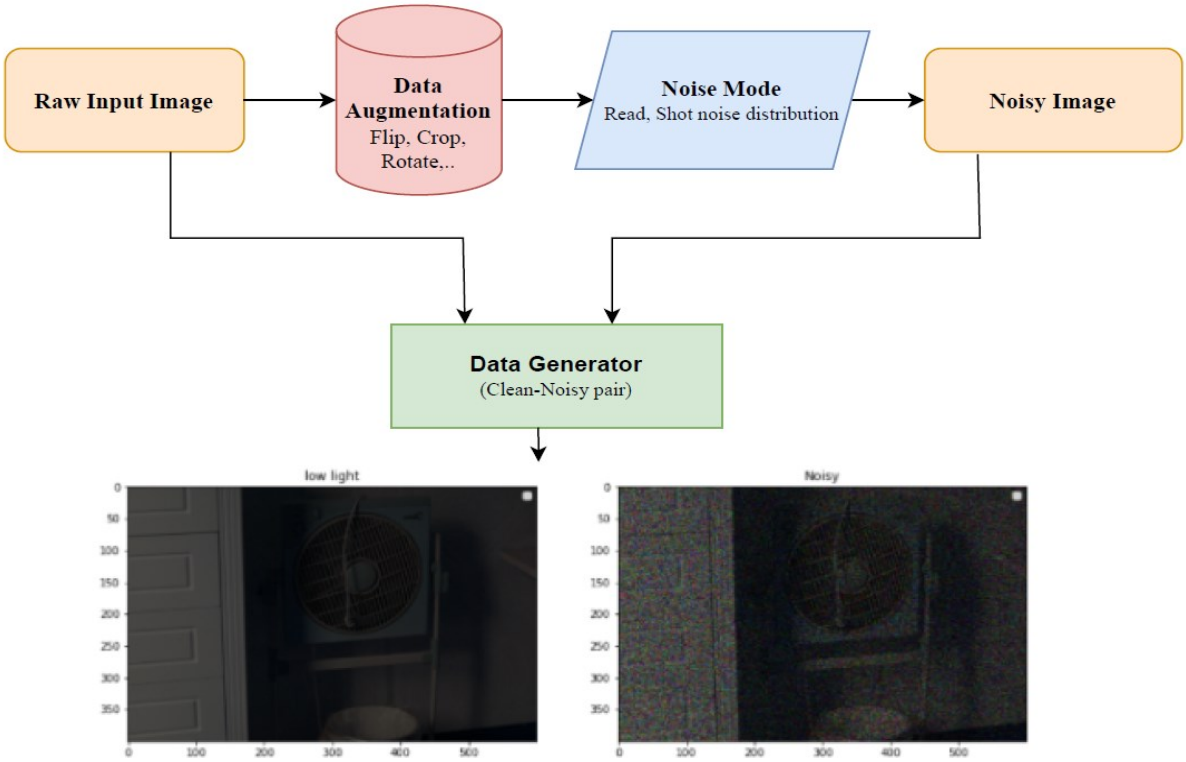


Figure 3.4: Data processing pipeline

4

Residual Learning

4.1 Residual Learning in Deep Learning

Residual Learning, a.k.a residual network or ResNet, is a technique used in deep learning to address the problem of vanishing gradients. Traditional deep networks are designed to learn a mapping function F(x) from the input data x to the output data y. However, the gradient of the loss function becomes very small or even vanishes as the network goes deeper, so it can be challenging for the network to learn a direct mapping between the input and output, which can lead to the network becoming harder to optimize and slowing down learning.

The problem of **gradient vanishing** in deep learning arises when the gradients in the backpropagation algorithm (shown in Fig. 4.1) become very small as they are propagated back through many layers of the neural network. This can make it difficult to train the network effectively because the weights are updated based on these gradients and if they are too small, the updates will also be too small and the training process will be slow or may even stall.

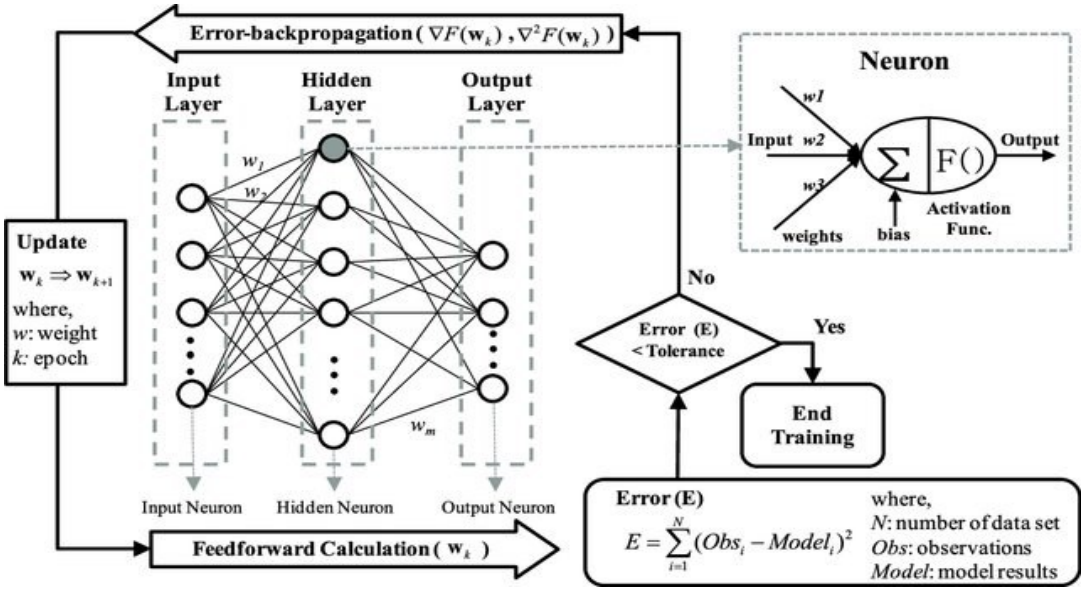


Figure 4.1: Schematic diagram of backpropagation training algorithm [52]

The reason why this occurs is that the backpropagation algorithm involves multiplying many small gradients together as they are propagated backward through the layers of the network. As the number of layers in the network increases, the gradients can become extremely small and may eventually approach zero, which means that the weights are no longer being updated effectively.

There are several ways to address the problem of gradient vanishing in deep learning, including using specialized activation functions (such as the rectified linear unit or ReLU), batch normalization, and techniques such as **residual connections** and **skip connections**, which allow for gradients to bypass certain layers and flow more easily through the network. This work is proposed mainly based on these techniques

Residual learning (Fig. 4.2) introduces skip connections to overcome the gradient vanishing problem, which allows the output of one layer to be added directly to the input of a following layer, and hence the model can learn to make residual corrections to the output of a previous layer, rather than trying to learn the entire transformation from scratch.

By allowing information to flood directly from earlier layers to subsequence layers, ResNets allow the model to be trained to greater depths and achieve higher accuracy than traditional deep neural networks, which can result in crucially better performance on a wide range of computer vision tasks.

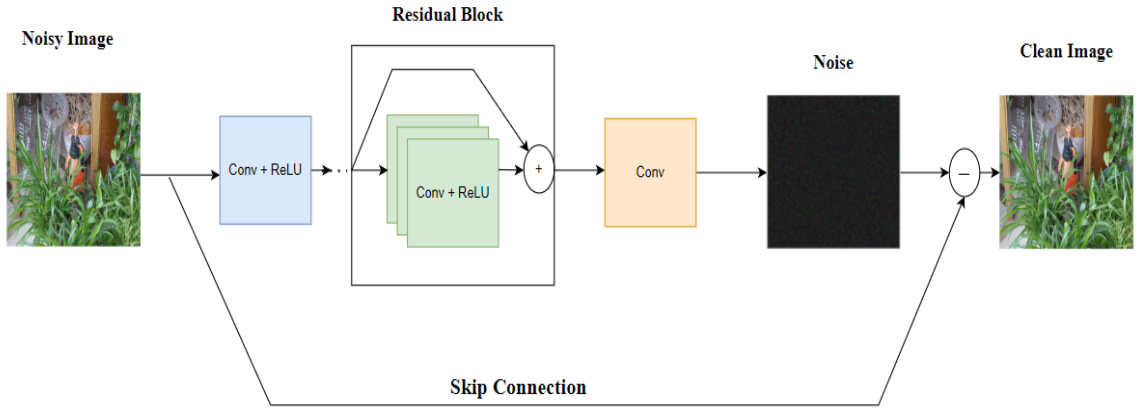


Figure 4.2: Residual learning

4.2 Residual Learning Formulation

As mentioned before, Residual learning [53] involves adding skip/short connections to a neural network. These connections allow the network to learn residual mappings, which are the difference between the desired output and the current output of the network.

The residual learning formulation is based on the idea that it is easier to learn the difference between the desired output and the current output of a network than to learn the entire mapping from input to output.

Mathematically, the residual learning formulation is expressed as follows:

Given an input $\boldsymbol{x}$, a residual block computes the output y as:

$$\boldsymbol{y = \mathcal{F}(x) + x} \quad (7)$$

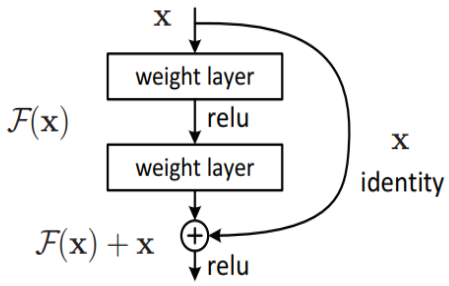


Figure 4.3: Residual block [53]

where $\boldsymbol{x}$ is the input to the layer, $\boldsymbol{\mathcal{F}(x)}$ is the residual function learned by the layer, and $\boldsymbol{y}$ is the output of the layer. Adding x directly to the output of the layer creates a "shortcut connection", which allows the gradient to flow easily during the training, so the network can learn to predict the residual function rather than the full mapping function. This formulation of residual learning using skip connections has become a popular approach in deep learning and has been widely used in a variety of applications, including image recognition, speech recognition, and natural language processing.

4.3 Networks Architecture

UNet [54] and ResUNet [53] are convolutional neural network architectures that were designed based on the concept of residual learning. The UNet design primarily uses skip connections, whereas ResUNet used many residual blocks that are stacked on top of each other, and each block has multiple convolutional layers and a shortcut/skip connection, to achieve the goal of alleviating the vanishing

gradient issue in deep networks and offering models that can be trained to greater depths with higher accuracy on a variety of tasks.

4.3.1 UNet Architecture

The U-Net [54] is a U-shaped encoder-decoder network structure, it considers an improvement of the existing fully convolutional networks. It consists of encoder blocks and decoder blocks connecting via a bridge. The encoder part halves the spatial dimensions of the input and doubles the number of filters (feature channels) at each encoder block. Likewise, the decoder part doubles the spatial dimensions and halves the number of feature channels. Fig. 4.4 shows Unet architecture.

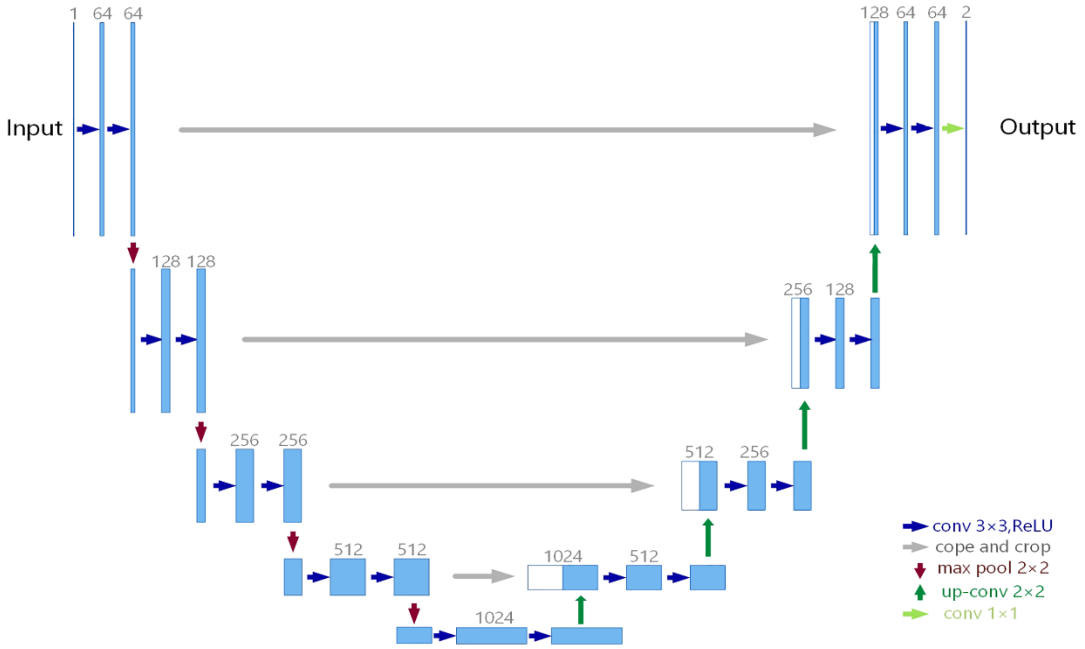


Figure 4.4: UNet architecture [53]

The **encoder Network** in UNet network usually consists of convolutional layers with stride 2, which reduces the spatial resolution of the input image by a factor of 2 in each layer with the aim of capturing high-level features of the image and reducing its dimensionality, offering more efficient processing in later layers by maintaining the spatial information of the image as it passes through the network. The encoder block is typically followed by a **bottleneck layer**, which is a set of convolutional layers that further reduce the spatial resolution of the image while preserving the number of feature maps. The bottleneck layer is important for capturing the most important features of the input image before passing them to the decoder block. The Encoder network represents a feature extractor in the UNet model it mainly learns the representation of the input image through a sequence of

encoder blocks, each block consists of two 3x3 convolution layers followed by the ReLU activation function which introduces a non-linearity into the network, and helps in better generalization of training data. The activation function followed by 2x2 max-pooling where the spatial dimensions of feature maps are reduced by half. This decreases the number of trainable parameters and thus reduces the computational cost.

The **skip connections** afford additional information that helps the decoder to generate better features. They act as a shortcut connection that helps in better flow of gradient during backpropagation which in turn helps the network learns the better representation for the input image.

The **decoder network** starts with a 2x2 transpose convolution. Next, it is concatenated with the corresponding skip connection feature map from the encoder block. These skip connections provide features from earlier layers that are sometimes lost due to the depth of the network. After that, two 3x3 convolutions are used, where each convolution is followed by a ReLU activation function. The output of ReLU function passes through 1x1 convolution with sigmoid activation which give the final output.

However, different variations and modifications of the UNet architecture have been proposed since then, with different numbers of encoder and decoder blocks. For example, some implementations may have more or fewer levels of resolution or may use different types of pooling or upsampling layers.

In practice, the number of encoder and decoder blocks is often chosen based on the complexity of the input data and the desired level of accuracy.

The key feature of the U-Net architecture is the usage of skip connections between the encoder and decoder. These skip connections allow the decoder to access feature maps from earlier layers in the encoder, which helps to preserve spatial information besides alleviating the gradient vanishing problem.

4.3.2 ResUNet Architecture

ResUNet [53] is a deep learning architecture that combines two popular neural network architectures, ResNet and U-Net. ResNet [30] is a convolutional neural network (CNN) that uses residual connections to improve training and performance, and the previously mentioned U-Net that uses an encoder-decoder architecture.

ResUNet, therefore, has an encoder based on the ResNet architecture, and a decoder based on the U-Net architecture. The encoder network consists of several blocks of convolutional layers with **residual connections**. Residual connections help in extracting feature maps from the input image. The decoder network, on the other hand, has a bilinear upsampling block that incrementally increases the spatial resolution of the feature maps.

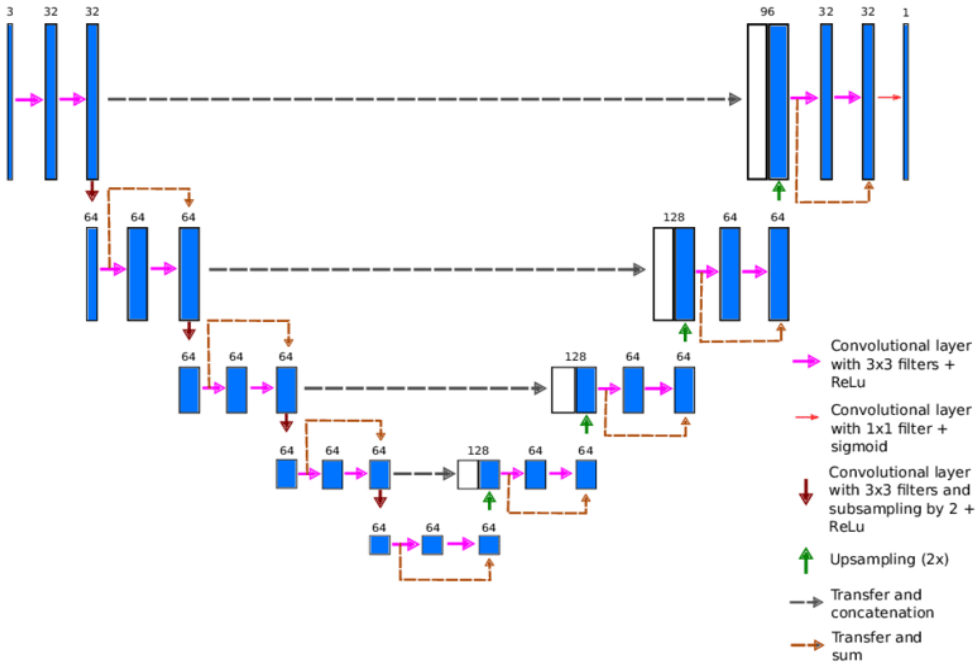


Figure 4.5: ResUNet architecture [55]

As Fig. 4.4 shows, we can notice that it has the same structure as UNet in Fig.4.5 with the addition of the residual connections within each block, which allow the network to better propagate gradients through the network during training. The combination of convolutional layers and residual connections leads to better computational efficiency compared to UNet since the residual connections reduce the trainable parameters that need to be learned by the network.

Overall, the UNet and ResUNet architectures have proven to be highly effective for a variety of computer vision tasks and have achieved state-of-the-art results on various benchmarks.

5

Proposed Denoising Model

This study introduces a deep learning approach for improving low-light images by removing noise and enhancing image quality. The method is divided into two stages: Denoising and Enhancement. The evaluation is performed using specific performance metrics, and the outcomes of the approach are presented in detail. The focus of this chapter is on the first stage, which is image denoising.

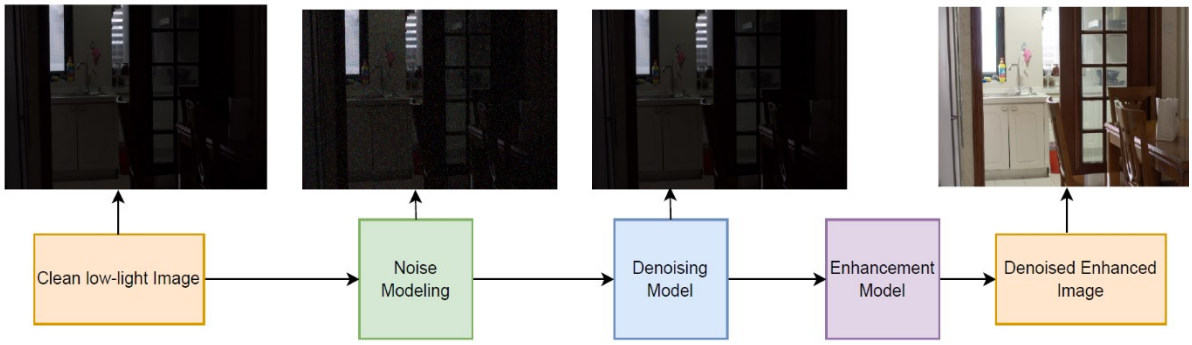


Figure 5.1: Flowchart of low-light image denoising and enhancement

5.1 Methodology

5.1.1 ResUnet Model

Our proposal is the deep ResUnet, a neural network that merges the UNet and residual neural network's strengths. By doing so, we gain two advantages: firstly, the use of residual units aids in training the network, and secondly, the skip connections present both within a residual unit and between low and high levels of the network enable efficient information propagation without degradation. This design results in a neural network requiring significantly fewer parameters while achieving comparable or superior performance on image denoising.

The denoising model is mainly composed of three parts: encoding, bridge, and decoding as shown in Fig. 5.2. A custom data generator is used to feed the model with a pair of clean low-light and noisy images, and all of these parts are

built with residual units of two 3x3 convolutional blocks added to the shortcut connection, where each convolutional block represents a combination of a BN layer, a ReLU activation layer, and a convolutional layer. The residual connections connect the input of one layer with the output of the following layer, and each residual connection has a 3x3 convolutional layer followed by a BN layer. Finally, the skip connections combine the feature extracted by the encoder block with the spatial information of the corresponding decoder block.

The first part is **the encoder**, which in turn handles learning features of the noisy image. It consists of four residual units, in each unit a stride of 2 is applied to the first convolutional block instead of using the max pooling operation to reduce the feature map by half as in the UNet model. In contrast, **the decoder** also composes of four residual units but before each unit, there is an up-sampling operation applied to the feature maps from the lower level and the concatenation with the feature maps from the corresponding encoding level.

At the last level of the decoding path, a 1x1 convolutional layer is applied to project the multi-channel feature maps into the desired output. The middle part serves as a **bridge** connecting the encoding and decoding paths. Table5.1 presents the convolutional layer and the output size after each residual unit.

	Unit Level	Conv. Filter	Stride	Output size
Input				400x600x3
Encoder	Level1	3x3/16	1	400x600x16
		3x3/16	1	400x600x16
	Level2	3x3/32	2	200x300x32
		3x3/32	1	200x300x32
	Level3	3x3/64	2	100x150x64
		3x3/64	1	100x150x64
	Level4	3x3/128	2	50x76x128
		3x3/128	1	50x76x128
Bridge		3x3/256	1	25x38x256
		3x3/256		25x38x256
Decoder	Level5	3x3/128	1	50x76x128
		3x3/128	1	50x76x128
	Level6	3x3/64	1	100x150x64
		3x3/64	1	100x150x64
	Level7	3x3/32	1	200x300x32
		3x3/32	1	200x300x32
	Level8	3x3/16	1	400x600x16
		3x3/16	1	400x600x16
output		1x1/3	1	400x600x3

Table 5.1: Convolutional layers in residual units of denoising model

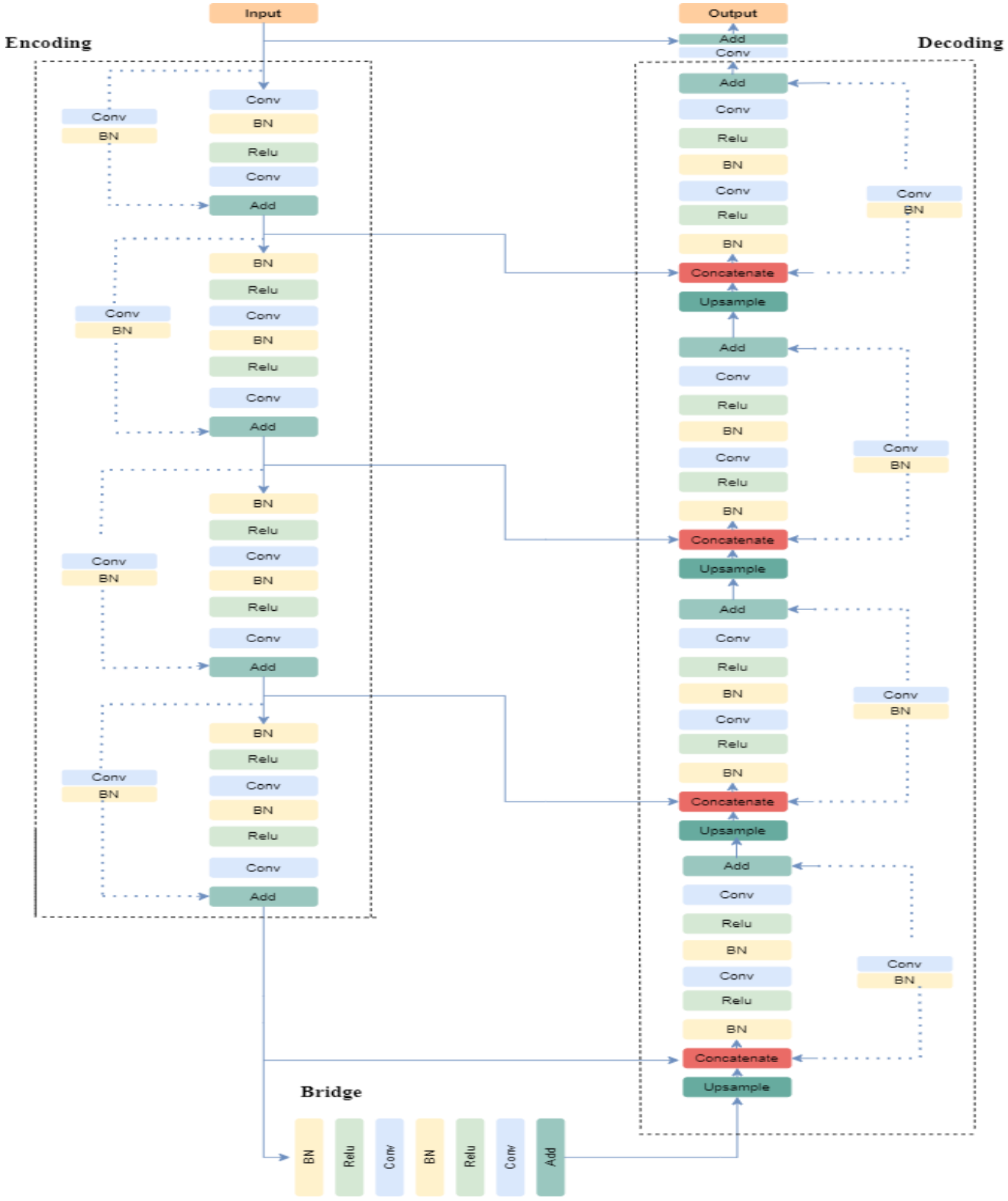


Figure 5.2: Architecture of the proposed denoising network

5.1.2 Loss Functions

Consider a clean image x corrupted by a generic noise v: $\boldsymbol{y} = \boldsymbol{x} + \mathrm{v}$. Our goal is to estimate the trainable parameters $\boldsymbol{W}$ of the network, such that it recovers the original clean image x. This is achieved by minimizing the loss between the generated noisy image and the original one. The problem of recovering x can be formulated as:

$$\hat{\boldsymbol{x}} = \boldsymbol{\mathcal{F}}(\boldsymbol{y}, \boldsymbol{W}) \tag{8}$$

where $\hat{\boldsymbol{x}}$ is the estimated recovered image of the original image $\boldsymbol{x}$, $\boldsymbol{y}$ is the corrupted image, and $\boldsymbol{W}$ is the network parameters.

Since the noisy image $\boldsymbol{y}$ contains most of the structure of the clean image $\boldsymbol{x}$ it makes sense to keep this structure and estimate only the noise for this reason residual learning is used.

Accordingly, in order to evaluate the trainable parameters $\boldsymbol{W}$, we can use the following optimization problem:

$$\boldsymbol{W}^* = \boldsymbol{argmin}\ \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}} \boldsymbol{\mathcal{L}}\left(\boldsymbol{\mathcal{F}}(\boldsymbol{y_i}, \boldsymbol{W}), \boldsymbol{x_i}\right) + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \| \boldsymbol{W} \| \tag{9}$$

Where $\{(\boldsymbol{y_i}, \boldsymbol{x_i})\}$ is the training dataset, $\boldsymbol{x_i}$ is the clean image and $\boldsymbol{y_i}$ is the corresponding noisy image. The first term of the Eqn. (7) corresponds to the fidelity term and the second one is the regularization term, while the hyperparameter $\lambda > 0$ controls the tradeoff between the two terms. The $\boldsymbol{\mathcal{L}}(y, x)$ is the loss function, in our case, the mean square error provided a better loss, i.e, we use the following function:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}) = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}} \|\boldsymbol{y} - \boldsymbol{x}\|^{\mathbf{2}} \tag{10}$$

5.1.3 Evaluation Metrics

There are many performance metrics used to evaluate these models and prove their worth or failure in achieving their endeavor of them. In this work, we chose **PSNR** and **SSIM** metrics for assessing the proposed models.

PSNR stands for Peak Signal-to-Noise Ratio and is a widely used metric in image processing to evaluate the quality of an image compared to its original or reference image. It measures the ratio of the maximum possible value of the signal to the noise in the image. The higher the PSNR value, the less noise there is in the processed image, and the better the image quality.

PSNR is calculated by comparing the mean square error (MSE) between the original and processed images to the maximum possible pixel value in the image. The formula for PSNR is:

$$\boldsymbol{PSNR = 20\ log10(MAXp) - 10\ log10(MSE)} \quad (11)$$

where **MAXp** is the maximum possible pixel value (usually 255 for 8-bit images), and **MSE** is the mean squared error between the original and processed images.

PSNR is a useful metric for evaluating the effectiveness of image compression algorithms or image processing techniques. However, it should be noted that PSNR does not always reflect the perceptual quality of an image, as some noise may not be noticeable to the human eye. Therefore, we use another metric, SSIM [56] stands for Structural Similarity Index, which is another widely used metric for evaluating the similarity between two images. Unlike PSNR, SSIM takes into account the structural information of the images and how the human visual system perceives the differences between the two images.

SSIM works by comparing the luminance, contrast, and structure of the original and processed images. The SSIM index ranges from -1 to 1, where a value of 1 indicates perfect similarity between the two images, and a value of -1 indicates complete dissimilarity. A value of 0 indicates that the two images are uncorrelated. The formula for SSIM is as follows:

$$\boldsymbol{SSIM(x,y) = [l(x,y)]^{\alpha}.[c(x,y)]^{\beta}.[s(x,y)]^{\gamma}} \quad (12)$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the two images being compared, $\boldsymbol{l(x,y)}$ represents the luminance similarity, $\boldsymbol{c(x,y)}$ represents the contrast similarity, and $\boldsymbol{s(x,y)}$ represents the structural similarity.

The parameters **α**, **β**, and **γ** are constants that control the relative importance of each component. Typically, they are set to **α = β = γ = 1**

It is worth noting that SSIM, like any other metric, has its limitations and may not always accurately reflect human perception. Therefore, the conjunction between PSNR and SSIM provides a more complete evaluation of image quality.

5.2 Experimental setup

5.2.1 Dataset

In the denoising model we use Low-Light (LOL) Dataset originally proposed for Unsupervised Low-light Image Enhancement with Decoupled Networks [32]. LOL is composed of 500 low-light and normal-light image pairs and is divided into 400 training pairs, 85 validation pairs, and 15 testing pairs. The Data generator augments the data and generates a random noise according to Eqn. (6) that simulates the noise generated by the camera's sensor and produces a pair of clean and noisy images to be fed to the model. All the images have the full resolution of 400 × 600 as described in Table 5.1.

5.2.2 Implementation details

The proposed model is implemented using Keras [33] framework. Adam optimizer [34] is used to optimize the whole network with learning rate (0.0001) to train the model.

We train the model on NVIDIA GTX1070 GPU with 8 Gb of Ram, we choose a batch size of 2 for 500 epochs. We use a small batch size to fit the GPU memory.

Overall, the training process involves to minimize the mean square loss (MSE) defined in Eqn. (10), and achieving the best values for the evaluation metrics (PSNR, SSIM).

5.3 Denoising Results

Before choosing the final loss function to train our model, we try out multiple loss functions during model training and compare the resulting performance to determine the most suitable one for our specific task. Table. 5.2 shows a comparison between the results of these functions. The comparison is done over 100 epochs.

Metric	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
Loss function	**MSE**		**MAE**		**Vgg Perceptual loss**		**MAE + MSE**		**MSE + MAE + Vgg**	
Values	**37.91**	**0.875**	**37.341**	**0.871**	**36.39**	**0.865**	**37.199**	**0.873**	**33.834**	**0.740**

Table 5.2: Comparison between different loss functions

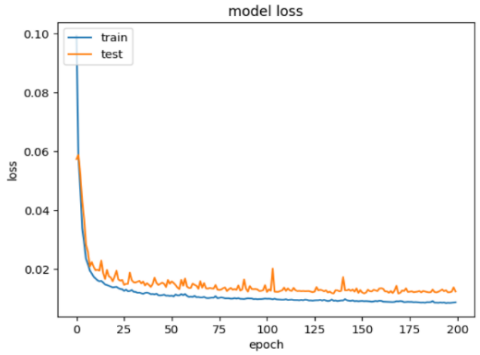
Where the **MAE** (Mean Absolute Error) is the average of the absolute differences between the predicted denoised image and the clean low-light image.

$$\mathcal{L}(y, x) = \frac{1}{N} \sum_{i=1}^{N} |y - x| \quad (13)$$

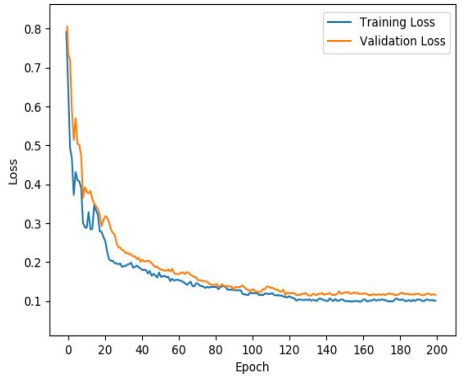
While the **Vgg perceptual loss** [57] is a type of loss function used in image processing tasks such as image style transfer, super-resolution, and image generation. It is named after the VGG-19 [58] network a popular convolutional neural network architecture used in computer vision tasks.

The VGG perceptual loss measures the difference between the feature representations of two images, where one image is a denoised image and the other is a target clean image. The feature representations are obtained by passing the images through a pre-trained VGG-19 network, and extracting the output of certain intermediate layers. By comparing the feature representations of the denoised image and the target image, the VGG perceptual loss encourages the denoised image to have similar low-level and high-level features as the target image.

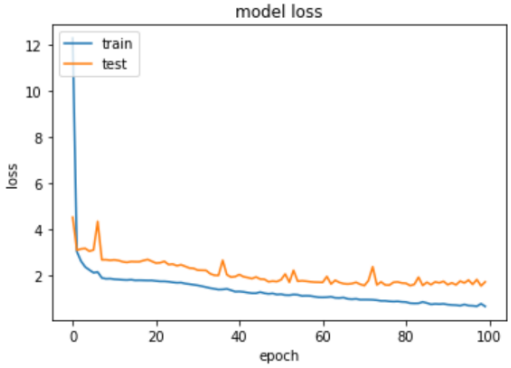
We also utilize a weighted sum to combine **multiple losses** in our approach. Fig. 5.3 illustrates the loss curves of these functions.



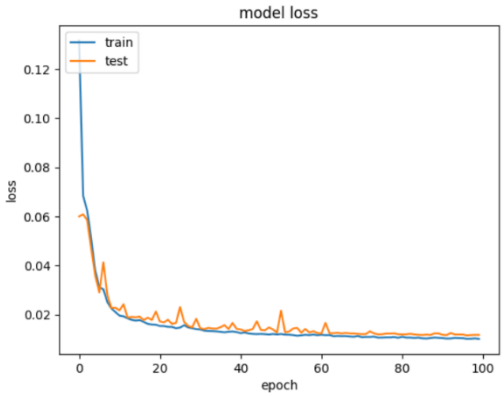
MAE



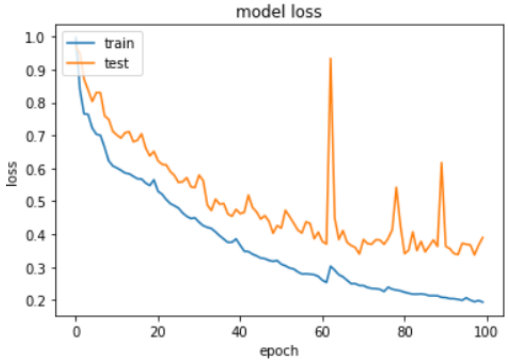
MSE



Vgg



MAE + MSE



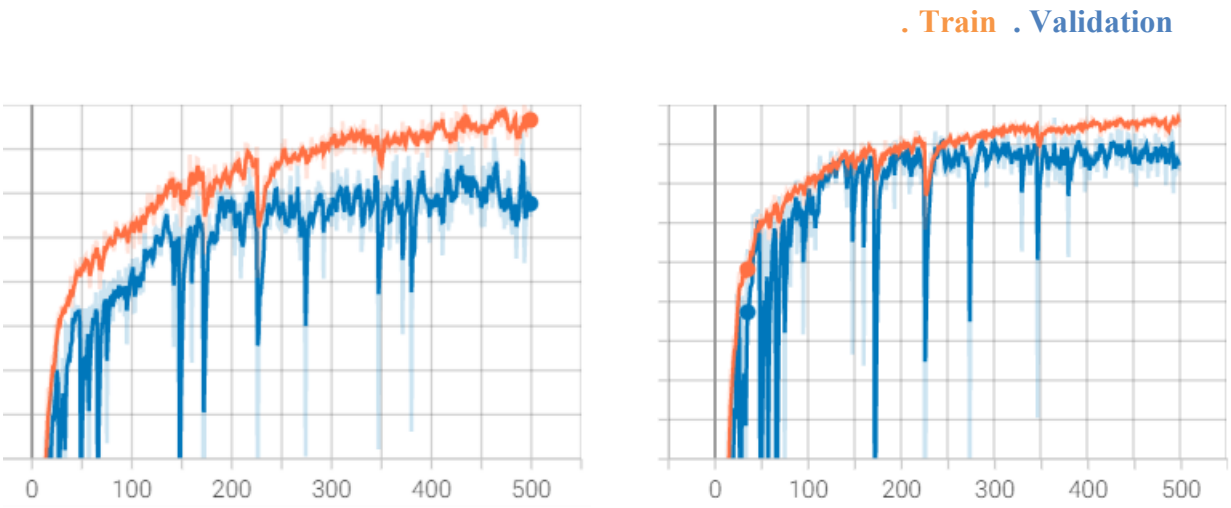
Weight sum of MSE, MAE and Vgg

Figure 5.3: Loss curves

It is evident that MSE yielded a satisfactory loss curve and the highest evaluation metrics value. The MAE also produced a favorable loss curve, however, it failed to deliver the intended output. Conversely, the Vgg loss curve seems to be indicating overfitting (for more epochs), while the combined losses resulted in a blurry output and lower metric values when compared to MSE. we chose **MSE** to train our proposed model.

Training the model for 500 epochs gave impressive results. It achieved PSNR = **44.85** and SSIM = **0.9576**.

The curves in the figures below show the improvement of evaluation metrics values and the decrease in the loss during training.



(a) (b)

Figure 5.4: (a) PSNR curve (b) SSIM curve

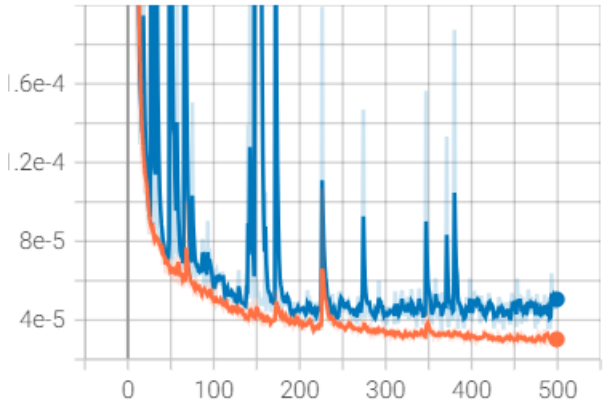


Figure 5.5: ResUnet MSE loss curve

Fig. 5.5 shows that after many epochs of training, the training and validation loss functions reach a minimum point of convergence and there is little difference between the two final values, the training data loss is slightly less than the validation data loss, indicating that we have a good fitting scenario for our model.

5.3.1 Sample Results

Ground-Truth ***Noisy-Image*** ***Denoised-output***

Figure 5.6: Test samples and denoised outputs

We can see from the resulting images that the output images are very near to the clean ground truth images, so we can conclude that the denoising phase was successful based on the outputs, evaluation metrics values, and loss curve.

Fig. 5.7 below shows some details of these denoised images compared to their noisy counterpart.

(a) Noisy Images (b) Denoised Image

Figure 5.7: Noisy and denoised samples

6

Enhancement Techniques

The objective of low-light enhancement in deep learning is to enhance the quality and visibility of images that were taken under low-light conditions. In such conditions, images can encounter challenges like low contrast, noise, and blurriness, which can make it challenging to extract meaningful information from them.

In this section, we will explore various methods to enhance image illumination, ranging from conventional techniques to deep learning techniques. The input to this model will be the image obtained from the denoising phase. We will cover histogram equalization, MIRNet [59] and Zero-dce [60] techniques.

6.1 Histogram Equalization

A **histogram** is a visual depiction that shows how the pixel intensities are distributed throughout an image. It counts the number of pixels that belong to different intensity ranges or bins to provide a summary of the image content.

Histogram equalization is a technique used in computer vision to enhance the contrast of an image by adjusting the distribution of its pixel intensities. The method works by re-mapping the pixel intensity values of an image so that they cover the entire range of values available.

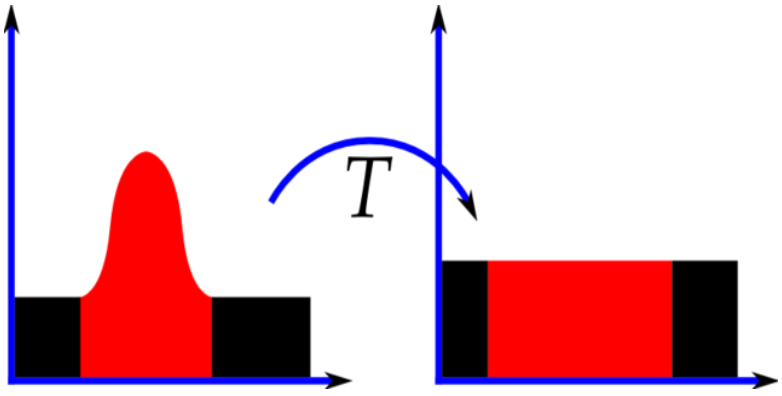


Figure 6.1: Histogram of an image before and after equalization [61]

The histogram equalization process involves calculating the cumulative distribution function (CDF) of the image's histogram and then scaling the pixel values to obtain a new histogram with a uniform distribution.

The CDF of an image's histogram is computed by summing the number of pixels in each intensity level and dividing by the total number of pixels. This function gives the probability of each intensity level occurring in the image. The pixel intensity values are then scaled based on the CDF values, so that higher intensity values are more spread out, and lower intensity values are compressed. Thus, the equalization function is:

$$\boldsymbol{s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r\,(r_j)} \tag{14}$$

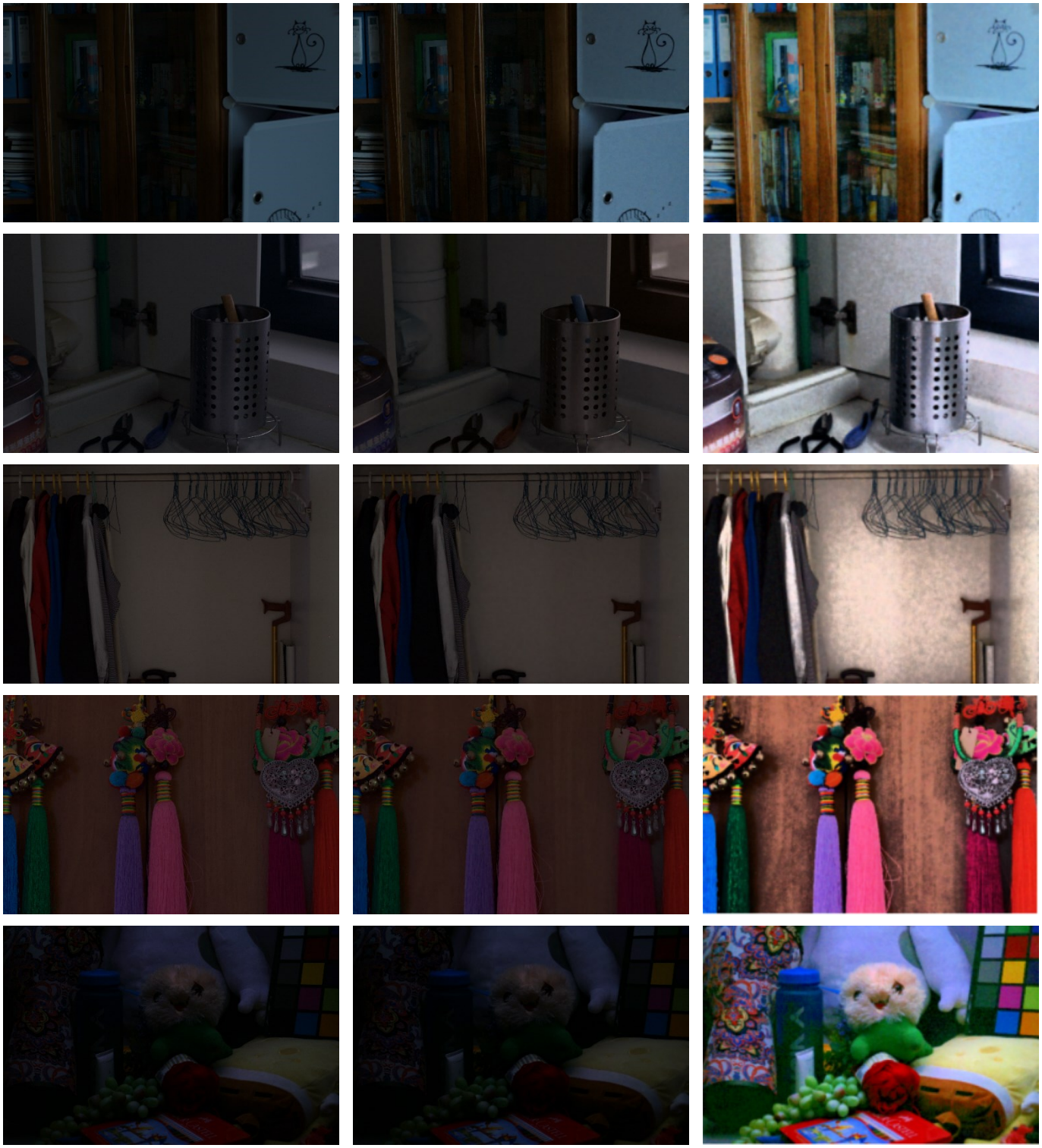
where **s** and **r** are the output and input pixel intensities respectively. **L** is the maximum intensity value (for n bit image **L** = 2^n). The probability of occurrence of the intensity level $\boldsymbol{r_j}$ in the image is approximated by:

$$\boldsymbol{p_r(r_j) = \frac{n_j}{MN}} \tag{15}$$

It is not advisable to separately apply histogram equalization to the red, green, and blue components of an image as it can result in significant changes to the image's color balance. Hence, to prevent this, we first converted the image to a different color space, such as HSV, before applying the equalization to the output of the denoising model (i.e., denoised images). This allowed the algorithm to be applied to the luminance channel without affecting the hue and saturation of the image.

6.1.1 Histogram Equalization Result

Clean Low-Light	*Denoised output*	*Enhanced output*



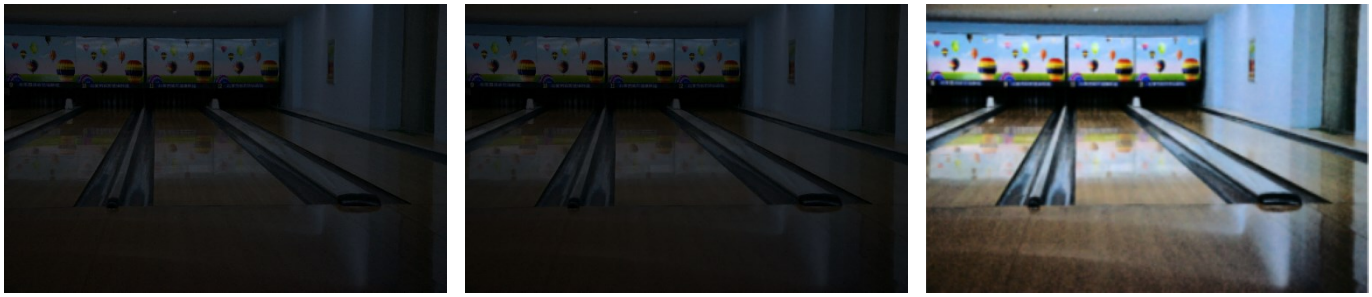


Figure 6.2: Histogram equalization enhanced results

In summary, our findings suggest that the use of histogram equalization can enhance an image's contrast, facilitating the differntiation of various objects or characteristics present in the image. Nevertheless, it may also cause some loss of details, as our experiment indicated that this method has the potential to amplify noise in the image since the histogram equalizer will redistribute the noisy pixels intensities for higher values.

6.2 Zero-Reference Deep Curve Estimation

This deep learning-based method is capable of handling various lighting conditions, including nonuniform and poor lighting situations. Instead of mapping images directly, the proposed technique reformulates the problem as an image-specific curve estimation task. To achieve this, the input low-light image is used to produce high-order curves that are then utilized to adjust the dynamic range of the input on a pixel-by-pixel basis. The curve estimation process is designed to preserve the image's range and contrast of neighboring pixels while being differentiable, allowing the adjustable parameters of the curves to be learned using a deep convolutional neural network. The proposed network [26] is lightweight, and it can be used iteratively to approximate higher-order curves to achieve more robust and accurate dynamic range adjustment.

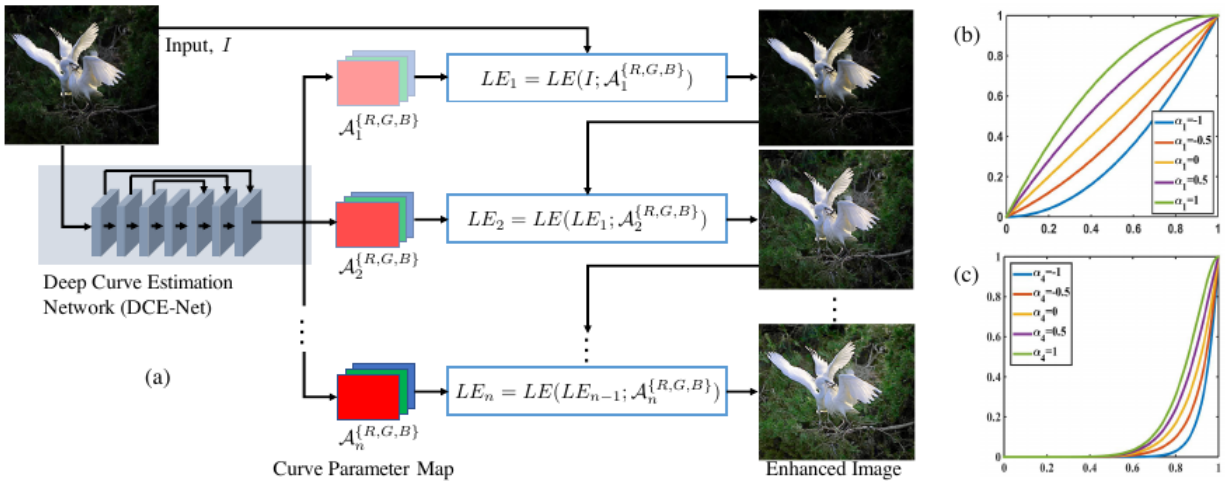


Figure 6.3: Framework of zero-reference deep curve estimation [60]

Fig. 6.3 illustrates the architecture of the Zero-Reference Deep Curve Estimation (Zero-DCE) framework, which uses a Deep Curve Estimation Network (DCE-Net) to estimate a set of Light-Enhancement (LE) curves that best fit the input image. The framework applies these curves iteratively to all pixels in the input's RGB channels to obtain the final enhanced image. This process involves key components, such as the LE-curve, DCE-Net, and non-reference loss functions, which will be detailed in the following sections.

6.2.1 Zero-DCE Framework

I. Light-Enhancement Curve (LEC) and pixel-wise Curve

The **LEC** represents a global adjustment of the brightness of the image, which is applied uniformly to all pixels in the image. It captures the overall illumination level of the image and adjusts it to a more suitable level for human perception. The LEC is learned by the Deep Curve Estimation Network (DCE-Net) and is applied to the input image as a single curve. It can be formulated as:

$$\boldsymbol{LE(I(x), \alpha) = I(x) + \alpha\,[I(x)(1 - I(x)]} \qquad (16)$$

where $\boldsymbol{x}$ denotes pixel coordinates, $\boldsymbol{LE(I(x), \alpha)}$ is the enhanced version of the given input $\boldsymbol{I(x)}$, and $\boldsymbol{\alpha \in [-1, 1]}$ is the adjustment trainable curve parameter.

On the other hand, the **pixel-wise** curve is a local adjustment of the brightness of the image that is applied on a per-pixel basis. It allows for fine-grained

adjustments of brightness at the local level, which can capture details and textures that may be lost in the LEC adjustment. The pixel-wise curve is also learned by the DCE-Net and is applied **iteratively** to each pixel in the image, and can be formulated as:

$$\boldsymbol{LE_n(x) = LE_{n-1}(x) + A_n\,[LE_{n-1}(x)(1 - LE_{n-1}(x)]} \qquad (17)$$

where **n** is the number of iterations, and $\boldsymbol{A_n}$ is the parameter map with the same size as the input image.

In summary, the LEC is a global adjustment that applies a single curve to the entire image, while the pixel-wise curve is a local adjustment that applies a curve to each pixel in the image. The combination of these two components allows the Zero-DCE approach to achieve accurate and fine-grained image enhancement in low-light conditions without the need for a reference image.

II. Deep Curve Estimation Network

The **Deep Curve Estimation Network** (DCE-Net) is a key component of the Zero-DCE approach. It is a convolutional neural network (CNN) that is trained to estimate a set of best-fitting Light-Enhancement (LE) curves, which are used to enhance the input image.

The DCE-Net takes the low-light image as input and outputs a set of LE curves, including the Light Enhancement Curve (LEC) and a pixel-wise curve. The DCE-Net is typically composed of seven convolutional layers. Each layer consists of 32 (3x3) convolutional kernels of stride 1 followed by ReLU activation function. The last convolutional layer is followed by Tanh activation to produce 24 parameter maps for 8 iterations (**n** = 8), where each iteration requires three curve parameter maps for the three channels. (Fig. 6.4).

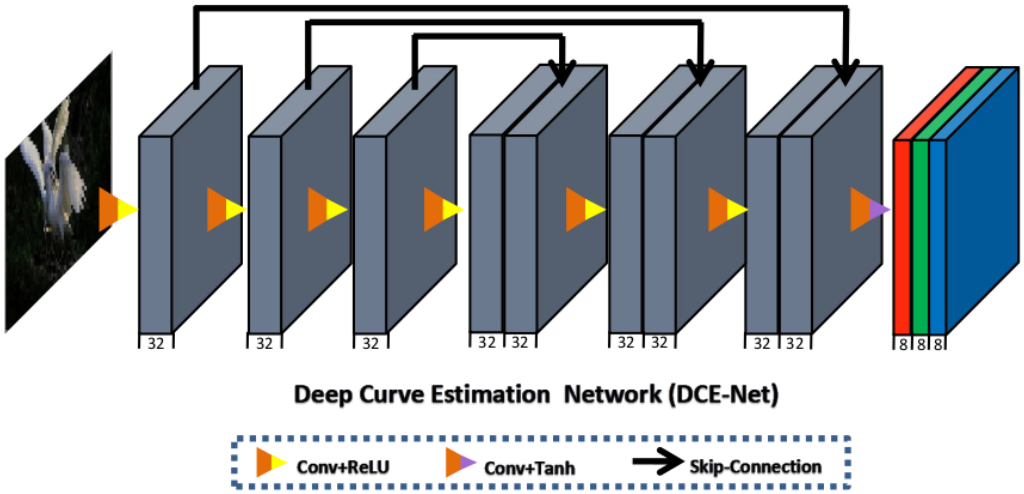


Figure 6.4: Deep curve estimation network architecture [62]

During training, it learns to estimate the LE curves that minimize the difference between the generated enhanced image and the ground truth image. This is typically done by minimizing a loss function (will be described in the next section).

The DCE-Net architecture can vary depending on the specific implementation but typically includes features such as skip connections to capture multi-scale information, batch normalization to reduce internal covariate shift, and residual blocks to improve gradient flow during backpropagation.

Overall, the DCE-Net is a powerful tool for enhancing low-light images by estimating the best-fitting LE curves, which are then applied to the image to produce a visually pleasing and enhanced output.

```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (Input Layer)         [(None, None, None,  0
_________________________________________________________________________________________________
conv2d (Conv2D)              (None, None, None, 3 896         input_1[0][0]
_________________________________________________________________________________________________
conv2d_1 (Conv2D)            (None, None, None, 3 9248        conv2d[0][0]
_________________________________________________________________________________________________
conv2d_2 (Conv2D)            (None, None, None, 3 9248        conv2d_1[0][0]
_________________________________________________________________________________________________
conv2d_3 (Conv2D)            (None, None, None, 3 9248        conv2d_2[0][0]
_________________________________________________________________________________________________
concatenate (Concatenate)    (None, None, None, 6 0           conv2d_3[0][0]
                                                              conv2d_2[0][0]
_________________________________________________________________________________________________
conv2d_4 (Conv2D)            (None, None, None, 3 18464       concatenate[0][0]
_________________________________________________________________________________________________
concatenate_1 (Concatenate)  (None, None, None, 6 0           conv2d_4[0][0]
                                                              conv2d_1[0][0]
_________________________________________________________________________________________________
conv2d_5 (Conv2D)            (None, None, None, 3 18464       concatenate_1[0][0]
_________________________________________________________________________________________________
concatenate_2 (Concatenate)  (None, None, None, 6 0           conv2d_5[0][0]
                                                              conv2d[0][0]
_________________________________________________________________________________________________
conv2d_6 (Conv2D)            (None, None, None, 2 13848       concatenate_2[0][0]
=================================================================================================
Total params: 79,416
Trainable params: 79,416
Non-trainable params: 0
_________________________________________________________________________________________________
```

Table 6.1: Zero-DCE layers

The statement highlights the fact that DCE-Net, a particular neural network architecture, has a relatively low number of trainable parameters, specifically 79,416. This characteristic makes it lightweight and capable of running on computational resource-limited devices, such as mobile platforms. In other words, DCE-Net is a computationally efficient model that can be easily deployed on devices with limited processing power without compromising its performance.

III. Non-Reference Loss Functions

In order to facilitate zero-reference learning within the Deep Curve Estimation Network (DCE-Net), a set of differentiable non-reference loss functions has been proposed. These loss functions enable the evaluation of the quality of the enhanced images that are generated by the DCE-Net.

The DCE-Net is trained using four different types of non-reference loss functions, which allow for the comparison of the generated images with their ground truth counterparts. These loss functions are designed to capture different aspects of image quality, including texture, color, and sharpness. The use of these loss functions ensures that the DCE-Net is trained to generate images that are visually pleasing and of high quality.

The **Spatial Consistency Loss (L_{spa})** is a loss function that promotes spatial consistency in the enhanced image by preserving the difference between neighboring regions in the input image and its corresponding enhanced version. The goal of this loss function is to ensure that the enhanced image maintains the same spatial coherence as the input image, which helps to preserve important details and structures in the image. It can be expressed as:

$$L_{spa} = \frac{1}{K} \sum_{i=1}^{K} \sum_{j \in \Omega(i)} (|Y_i - Y_j| - |I_i - I_j|)^2 \qquad (18)$$

a parameter **K** represents the number of local regions in the image. Each local region is centered at a specific point, and includes four neighboring regions - top, down, left, and right - denoted by $\Omega(i)$. Within each local region, the average intensity value of the corresponding region in the enhanced image is represented as **Y**, while the average intensity value of the same region in the input image is represented as **I**.

The **Exposure Control Loss (L_{exp})** is a loss function that is designed to limit the impact of under- or over-exposed regions in the image. This loss function measures the distance between the average intensity value of a local region and a desired exposure level, denoted as E. The well-exposedness level E is typically set to the gray level in the RGB color space, following existing practices [44,45]. In this experiment, we set E to 0.6. The formula for the Exposure Control Loss, denoted as L_{exp}, is used to optimize this aspect of the image enhancement process is expressed as:

$$L_{exp} = \frac{1}{M} \sum_{K=1}^{M} |Y_k - E| \qquad (19)$$

Where $\boldsymbol{M}$ represents the number of nonoverlapping local regions, and **Y** is the average intensity value of a local region in the enhanced image.

The **Color Constancy Loss** ($\boldsymbol{L_{col}}$) is designed to correct potential color deviations in the enhanced image, while also building relationships among the three adjusted color channels. This loss function is based on the Gray-World color constancy hypothesis [46], which suggests that the average color in each sensor channel should be gray over the entire image.

To achieve color constancy in the enhanced image, the Color Constancy Loss measures the difference between the average intensity values of the three-color channels (R, G, and B) in the enhanced image, and the average intensity value of the green channel in the input image as follow:

$$\boldsymbol{L_{col}} = \sum_{\forall(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{\epsilon}}(\boldsymbol{J^p} - \boldsymbol{J^q})^2 \quad \text{, where } \boldsymbol{\epsilon} = \{(\boldsymbol{R},\boldsymbol{G}),(\boldsymbol{R},\boldsymbol{B}),(\boldsymbol{G},\boldsymbol{B})\} \quad (20)$$

where $\boldsymbol{J^p}$ denotes the average intensity value of $\boldsymbol{p}$ channel in the enhanced image, $\boldsymbol{(p, q)}$ represents a pair of channels.

The Color Constancy Loss is an important component of the Zero-DCE network, as it helps to ensure that the enhanced image has accurate and consistent color representations.

The **Illumination Smoothness Loss** ($\boldsymbol{L_{ill}}$) is used to preserve the monotonicity relations between neighboring pixels in the image. This is achieved by adding an illumination smoothness loss to each curve parameter map **A** in the Zero-DCE network. It can be defined as:

$$\boldsymbol{L_{ill}} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{n=1}}^{\boldsymbol{N}} \sum_{\boldsymbol{c}\in\boldsymbol{\varepsilon}} \left(|\boldsymbol{\nabla_x A_n^c}| + |\boldsymbol{\nabla_y A_n^c}|\right)^2, \quad \text{where } \boldsymbol{\varepsilon} = \{\boldsymbol{R},\boldsymbol{G},\boldsymbol{B}\} \quad (21)$$

where $\boldsymbol{N}$ is the number of iterations, $\boldsymbol{\nabla_x}$ and $\boldsymbol{\nabla_y}$ represent the horizontal and vertical gradient operations, respectively.

The **total loss function** used to train the Zero-DCE network and guide the image enhancement process is defined as a combination of the four individual loss functions described earlier. The expression for the total loss function is given by:

$$\boldsymbol{L} = \boldsymbol{L_{spa}} + \boldsymbol{L_{exp}} + \boldsymbol{\omega_{col}}\boldsymbol{L_{col}} + \boldsymbol{\omega_{ill}}\boldsymbol{L_{ill}} \quad (22)$$

Here, $\boldsymbol{L_{spa}}$, $\boldsymbol{L_{exp}}$, $\boldsymbol{L_{col}}$, and $\boldsymbol{L_{ill}}$ represent the Spatial Consistency Loss, Exposure Control Loss, Color Constancy Loss, and Illumination Smoothness Loss,

respectively. The coefficients $\boldsymbol{\omega_{col}}$ and $\boldsymbol{\omega_{ill}}$ are hyperparameters that control the relative importance of the corresponding loss term and they are set to 0.5, and 20 respectively to balance the scale of losses, where $\boldsymbol{L_{col}}$ pays attention to the relations among three channels when curve mapping is applied. While $\boldsymbol{L_{ill}}$ emphasizes the correlations between neighboring regions. The importance of $\boldsymbol{L_{spa}}$ lies in preserving the difference between neighboring regions between the input and the enhanced image.

By minimizing the total loss $\boldsymbol{L}$ during training, the Zero-DCE network is able to learn the optimal set of curve parameters and produce high-quality enhanced images that are visually appealing and consistent with the underlying scene.

6.2.2 Zero-DCE Experiment and Results

I. Implementation Details

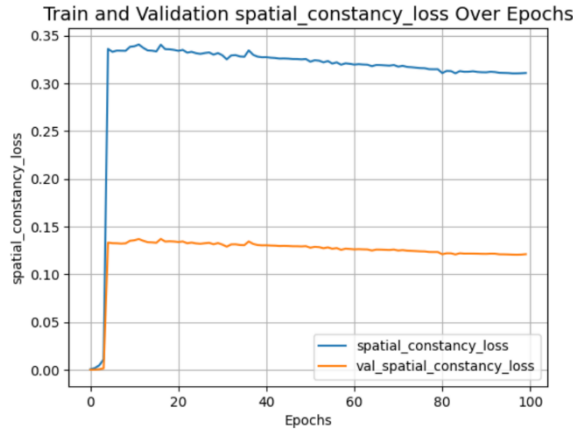
As previously mentioned, the input to this stage of the work is the denoised images obtained from the first denoising phase. The Zero-DCE model was implemented using the Keras framework with the Adam optimizer and a learning rate of 0.0001. The model was trained on an NVIDIA GTX1070 GPU with 8GB of RAM, and a batch size of 2 was selected for 100 epochs. A small batch size was used to ensure that the GPU memory was sufficient.

During the training process, the goal was to minimize the total loss function defined in Eqn. (20) while achieving the best possible values for the evaluation metrics such as PSNR and SSIM. By optimizing the model parameters in this way, we were able to produce high-quality enhanced images that effectively preserve the important details and characteristics of the original scene while eliminating noise and other unwanted artifacts.

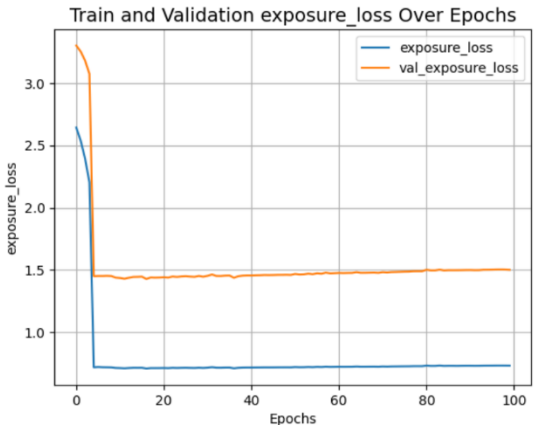
II. Results

Fig. 6.5 illustrates the curves of the loss functions obtained after training the model.

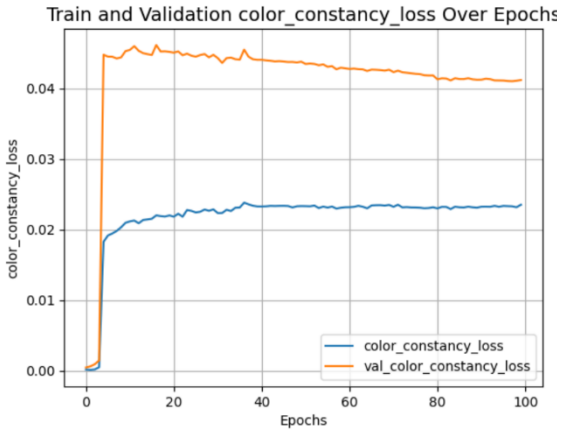
A. Losses Curve



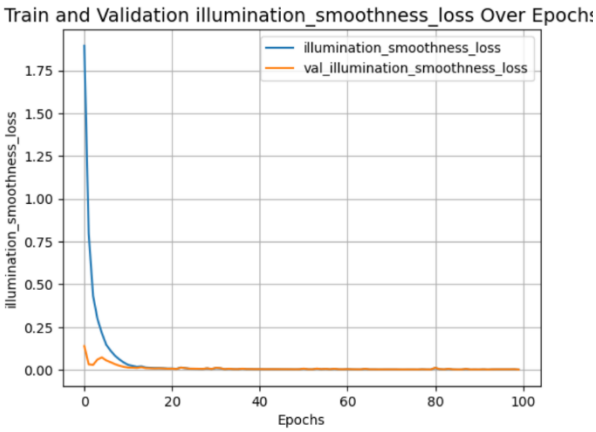
Spatial Constancy loss curve



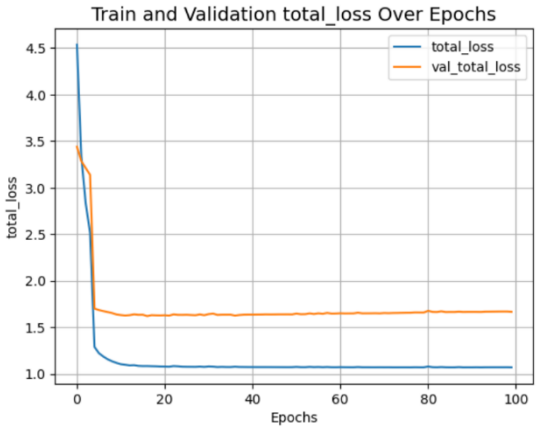
exposure loss curve



Color constancy loss curve



Illumination smoothness loss curve



Total loss curve

Figure 6.5: Zero-dce loss curves

The total loss curve in Fig. 6.5 is used to measure the difference between the enhanced image and the ground truth image. It is used to optimize the deep network during training. The curve estimation is specially designed, considering pixel value range, monotonicity, and differentiability. It can be seen in the total loss curve where the training loss continues to decrease while the validation loss starts to increase. This indicates that the model is starting to overfit the training data and will not generalizing well to new data.

B. Sample Results

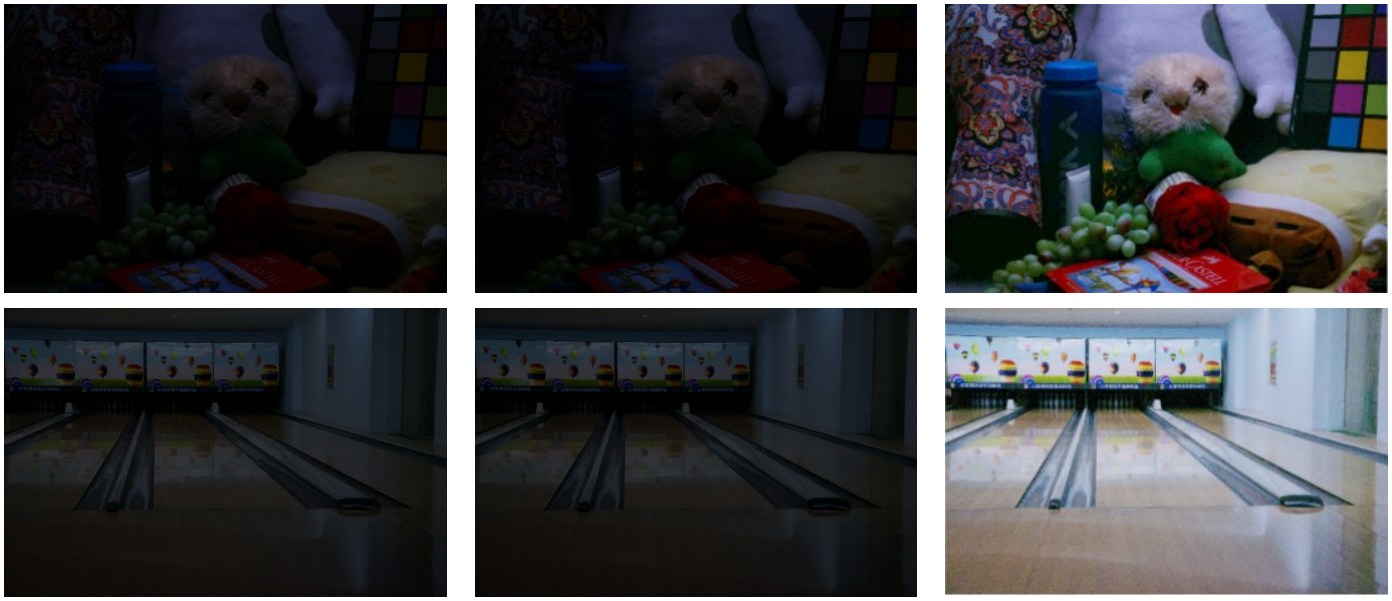


Figure 6.6: Zero-dce enhanced results

In the experiment, we discovered that Zero-DCE is appealing due to its relaxed reference picture assumption, i.e., it does not require any paired or even unpaired data during training. This is accomplished through the use of a set of carefully formulated non-reference loss functions, which indirectly measure the enhancement quality and drive the network's learning. Despite its simplicity, it demonstrates that it generalizes well to a variety of lighting situations. This approach is effective because image enhancement can be accomplished using an intuitive and simple nonlinear curve mapping technique.

6.3 Multi-Scale Information Retention Network

The Multi-Scale Information Retention Network (MIRNet) [59] is a convolutional neural network (CNN) that has been specifically designed for the low light image enhancement. The network's architecture is tailored to retain and enhance low light image details while suppressing noise and artifacts that may appear in the image. This involves increasing the resolution of a low-light image to produce a high-resolution version with greater clarity. MIRNet uses a multi-scale approach to retain important information at various levels of detail, which is achieved through its hierarchical architecture with multiple levels of abstraction. The input image is processed at different scales, using convolutional layers to

extract increasingly complex features, which are then combined to create a multi-scale representation of the original image. To prevent the vanishing gradient problem that can occur in deep networks, MIRNet also employs residual connections, which help the network to learn residual mappings between the low-light input and enhanced output images.

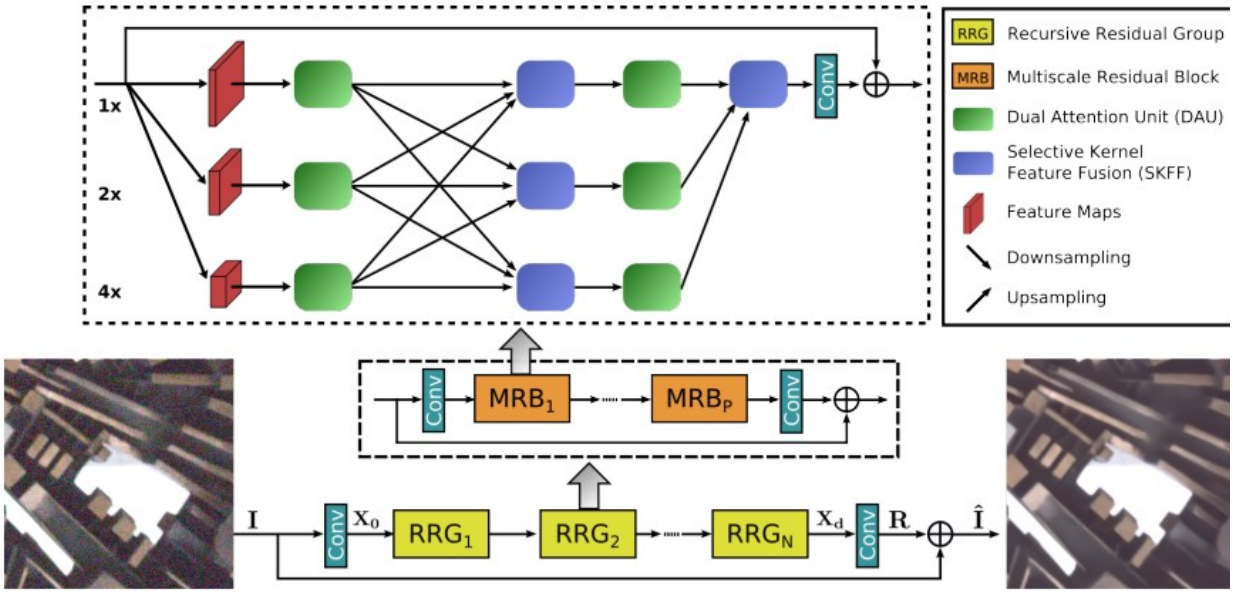


Figure 6.7: MIRNet architecture [59]

6.3.1 MIRNet Framework

MIRNet's architecture comprises three primary modules: a feature extraction module, a multi-scale feature fusion module, and a reconstruction module.

The **Feature Extraction Module** of MIRNet is designed to extract features from the input image at different scales. It is composed of a set of convolutional layers that operate on the input image at multiple scales, extracting progressively more complex features as the scale increases. This module plays a crucial role in enabling the network to capture both local and global information in the input image.

The **Multi-Scale Feature Fusion Module** of MIRNet takes the features extracted by the feature extraction module and fuses them together to create a multi-scale representation of the input image. This is done using a set of learnable weights that determine how much weight each scale should be given in the final representation.

The **Reconstruction Module** in MIRNet utilizes the multi-scale representation of the input image that is generated by the multi-scale feature fusion module to produce the high-resolution output image. This module

comprises a set of convolutional layers that operate on the multi-scale representation, followed by an upsampling layer that enhances the resolution of the output image. By employing this module, MIRNet can effectively reconstruct a high-quality image from the multi-scale features.

Apart from the aforementioned modules, MIRNet incorporates residual connections to mitigate the vanishing gradient problem that can arise in deep networks. These connections enable the network to learn residual mappings between the low-light input image and the output image. By leveraging residual connections, MIRNet can effectively capture important details of the input image and learn to produce high-quality output images.

The network first applies a convolutional layer to extract low-level features from the low-light input. The feature map then traverses N recursive residue groups (RRGs) to get deep features. Note that each RRG contains multiple multiscale residual blocks. Then we apply a convolutional layer to the deep features and get the residual image. Finally, the enhanced image is obtained by adding the residual image to the low-light input image.

The following are the major components of these modules:

I. Selective Kernel Feature Fusion

Two operations **Fuse** and **Select** perform the dynamic modification of receptive fields in the Selective Kernel Feature Fusion or SKFF module. The information from multiple-resolution sources is combined by the Fuse operator to produce global feature descriptors. These identifiers are used by the Select operator to recalibrate the feature maps before their aggregation.

Fuse operator: Three parallel convolution streams bearing various informational scales are inputs to the SKFF. We first combine these multi-scale features using an element-wise sum, and then across the spatial dimension, we use Global Average Pooling (GAP) to create a compact feature representation, we then apply a channel-downscaling convolution layer. This layer then goes through three parallel channel-upscaling convolution layers giving us three feature descriptors.

Select operator: Using the feature descriptors as input, this operator uses the softmax function to produce the corresponding activations, which are then used to adaptively recalibrate multi-scale feature maps. The product of the corresponding multi-scale feature and the feature descriptor is used to describe the aggregated features.

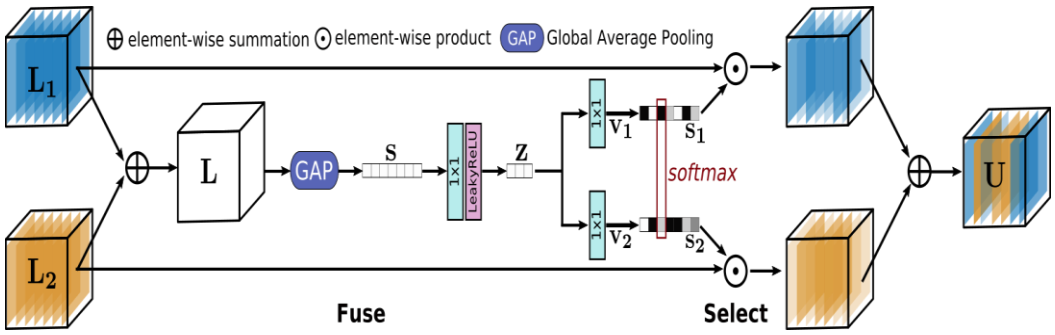


Figure 6.8: Selective kernel feature fusion [59]

II. Dual Attention Unit

Features in the convolutional sequences are extracted using the Dual Attention Unit, or DAU. The DAU block provides the means to share information within a feature tensor along the spatial and channel dimensions while the SKFF block fuses information across multi-resolution branches. Only more informative characteristics are allowed to continue after being suppressed by the DAU. Utilizing the Channel Attention and Spatial Attention processes, this feature calibration is accomplished.

By using squeeze and excitation procedures, the **Channel Attention** branch takes advantage of the inter-channel relationships of the convolutional feature maps. The squeeze operation takes as input a feature map, uses Global Average Pooling across spatial dimensions to encode global context, and produces a feature descriptor as a result. This feature description is subjected to two convolutional layers, sigmoid gating, and the excitation operator, which produces activations. Rescaling the input feature map using the output activations results in the output of the Channel Attention branch.

The inter-spatial correlations of convolutional features are intended to be taken advantage of by the **Spatial Attention** branch. To create a spatial attention map and use it to recalibrate the incoming characteristics is the aim of spatial attention. The Spatial Attention branch first independently performs Global Average Pooling and Max Pooling operations on input features along the channel dimensions, concatenates the outputs to form a resultant feature map, and then obtains the spatial attention map by passing the resultant feature map through convolution and sigmoid activation. The incoming feature map is then scaled using this spatial attention map.

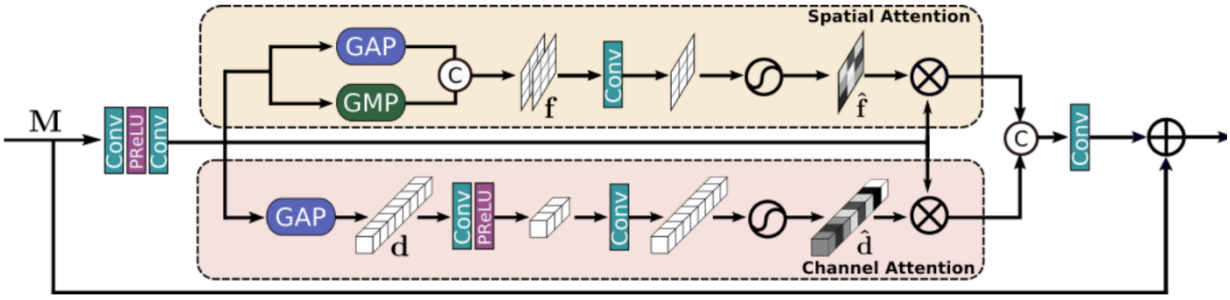


Figure 6.9: Dual attention unit [59]

III. Multi-Scale Residual Block

While keeping high-resolution representations and getting rich contextual data from low-resolutions, the Multi-Scale Residual Block can produce an output that is spatially precise. The MRB is composed of a number (three in this work) of parallel fully-convolutional streams. It enables information sharing between parallel streams so that low-resolution features can be used to help consolidate high-resolution features, and vice versa. A recursive residual architecture (with skip connections) is used by the MIRNet to facilitate information flow during the learning process. Residual resizing modules are used to carry out the downsampling and upsampling operations that are used in the Multi-scale Residual Block in order to preserve the residual character of our architecture.

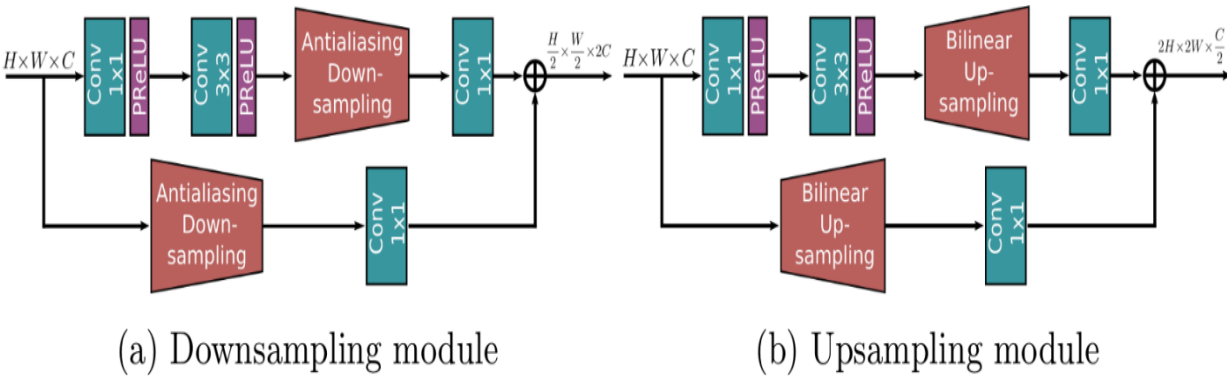


Figure 6.10: Multi-Scale residual block [59]

6.3.2 MIRNet Experiment and Results

I. Implementation Details

To prevent the requirement of training the Multi-Scale Information Retention Network (MIRNet) from scratch, a pre-trained model was utilized on our denoised output derived from the denoising phase. Given that MIRNet has a substantial number of trainable parameters, training it from the scratch can consume a considerable amount of computational resources. Therefore, the application of a pre-trained network could offer a more efficient method for achieving superior results on our dataset.

During training, the network is optimized to minimize the difference between the enhanced image and the ground-truth denoised image using Charbonnier Loss. Where Charbonnier loss [47] is a modification of the L1 loss that is more robust to outliers and can handle images with various noise levels effectively. The Charbonnier loss is defined as:

$$\mathbf{L}\left(\hat{\mathbf{I}}, \mathbf{I}^{*}\right) = \sqrt{\|\hat{\mathbf{I}} - \mathbf{I}^{*}\|^{2} + \boldsymbol{\epsilon}^{2}} \qquad (23)$$

Where $\hat{\boldsymbol{I}}$ is the enhanced image, $\boldsymbol{I}^{*}$ is the ground-truth image and $\boldsymbol{\epsilon}$ is a constant set to 10^{-3}. Adam optimizer [34] is used to optimize the whole network with learning rate (0.0001) to train the model.

II. Results

A. Loss Curve

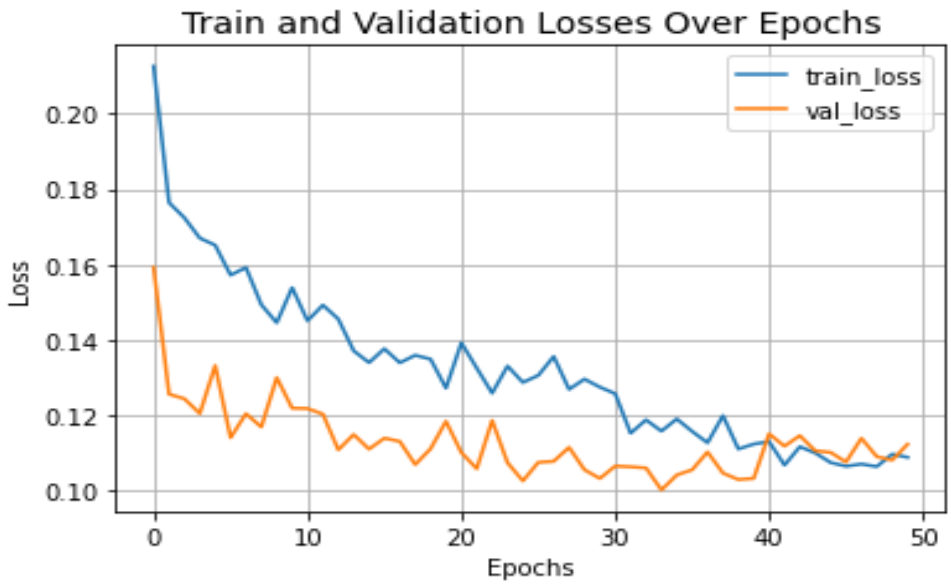


Figure 6.11: MIRNet loss curve

As shown in Fig. 6.11, the learning curve for training loss improves, as does the learning curve for validation loss. We can state that the validation loss is lower than the training loss at the start of training, indicating that the validation dataset may be easier for the model to predict than the training dataset.

B. Sample Results

Clean Low-Light ***Denoised output*** ***Enhanced output***

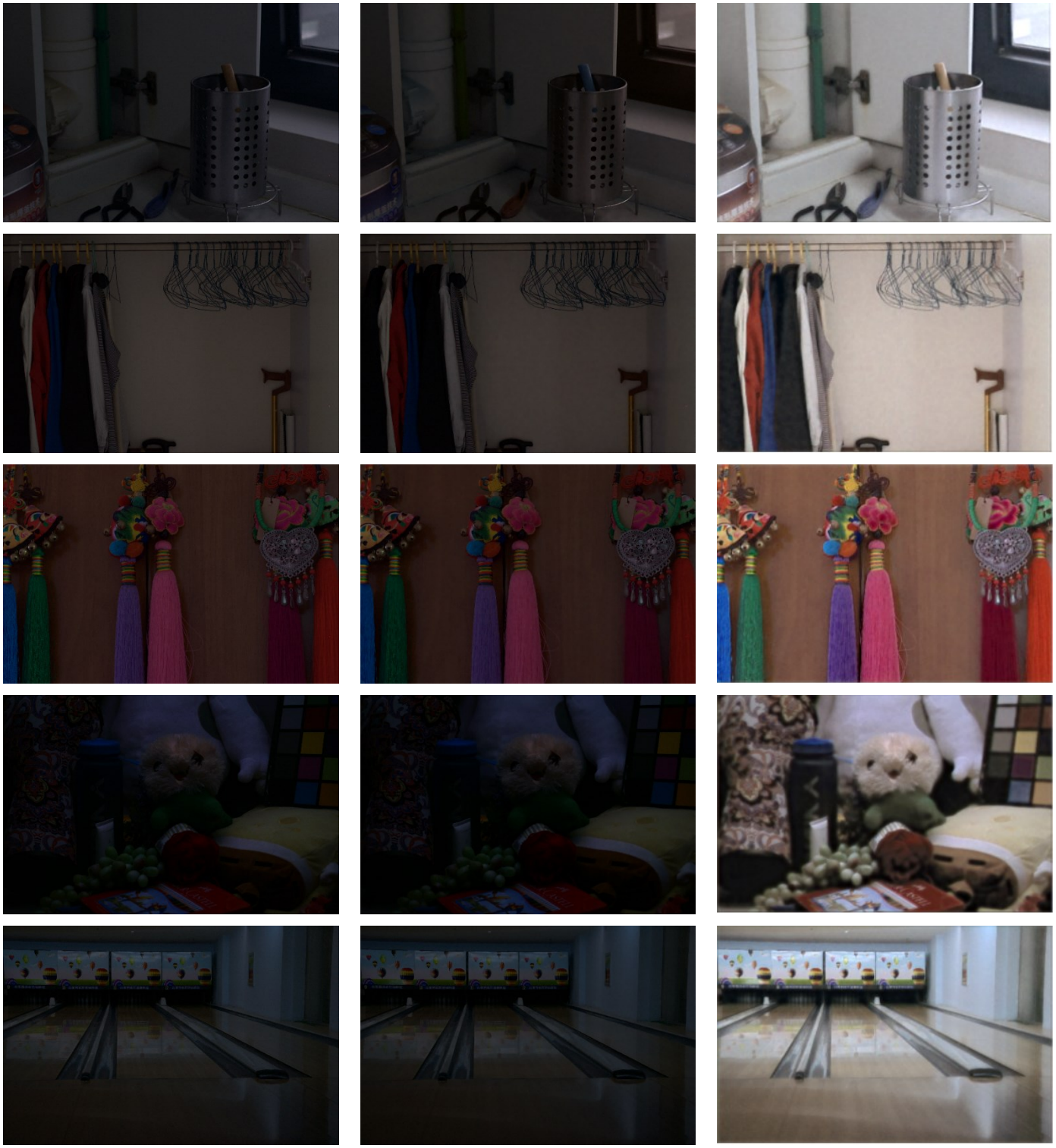


Figure 6.12: MIRNet enhanced results

As demonstrated by sample results, the MIRNet method produces enhanced images that are visually closer to the ground truth in terms of brightness and global contrast.

6.4 Results Comparison and Evaluation

6.4.1 Denoising phase:

(a) Low-Light Input (b) Noisy Low-light (c) Denoised Output

Figure 6.13: Final ResUnet output

Metrics	PSNR	SSIM
ResUnet	**44.85**	**0.9576**

Table 6.2: Evaluation metrics of ResUnet

Based on the results of the trained de-noising Model shown in Fig. 6.13 and table 6.2, we can conclude that the proposed model achieved its intended goal, as we obtained images as close as possible to the clean image, and we achieved high values of the evaluation metrics (PSNR,SSIM), which indicate that we get a better quality of the reconstructed image. These results will be used as input for the enhancement phase to obtain an image with higher brightness.

6.4.2 Enhancement phase:

(a) Input (Denoised Output) (b) Histogram Equalization (c) MIRNet (d) Zero-Dce

Figure 6.14: Visual comparison with enhancement methods

Metric	HE	MIRNet	Zero-Dce
PSNR	19.52	24.73	30.67
SSIM	0.701	0.835	0.923

Table 6.3: Quantitative comparison on LOL dataset in terms of PSNR, SSIM

According to Fig. 6.14 and Tabel 6.3, while histogram equalization brightens the image, it also highlights the presence of noise on it, whereas MIRNet recovers high-quality image content from its denoised low-light version by learning an enriched set of features that combines contextual information from multiple scales while preserving high-resolution spatial details. It enhances color and contrast adjustments, resulting in images that appear more natural and pleasant. Whereas zero-dce is efficient because low-light image enhancement is achieved through a straightforward and intuitive nonlinear curve mapping. Despite its simplicity and lightness, its results produced higher quality values, and it was considered the best of the bunch based on outcome samples and evaluation metrics.

7

Conclusion

The aim of this thesis was to eliminate the image noise that is produced by the camera sensor during the image-capturing process and enhance low-light images captured in low-light conditions. After reviewing some of the previous scientific literature, we investigate the use of residual learning in this work. Residual learning has shown remarkable performance in such tasks. It involves adding short/skip connections that enable the network to learn residual mappings, which are the difference between the desired output and the current output of the network. We explore architectures and loss functions that are designed for low-light image denoising and enhancement and prove their effectiveness by conducting extensive experiments on a benchmark dataset.

Regarding the denoising task, we integrate residual blocks into UNet in the ResUnet we investigate. This allows us to combine the advantages of these two networks at the same time. The ResUnet is trained on noisy low-light images to extract feature information and spatial information of the noisy image within the encoder-decoder network framework.

We used the denoised results as input to a new network to enhance low-light images and obtain clear details of an image. For this purpose, we employed a variety of techniques ranging from traditional methods like histogram equalization to deep learning models such as zero-dce, which estimates pixel-wise and high-order tonal curves for dynamic range adjustment of a given image using a lightweight CNN, and MIRNet works in a more complex way by learning an enriched set of features that combine contextual information from multiple scales while still preserving high-resolution spatial details. At the conclusion of this study, we briefly compared the performance of these models to their evaluation metric's actual values.

Bibliography

[1] Zhang, M., Gunturk, B.K.: Multiresolution bilateral filtering for image denoising. IEEE Trans. Image Process. 17(12), 2324–2333 (2008)

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

[3] Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Trans. Image Process. 16(8), 2080–2095 (2007)

[4] Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Trans. Image Process.15(12), 3736–3745 (2006)

[5] Rajwade, A., Rangarajan, A., Banerjee, A.: Image denoising using the higher order singular value decomposition. IEEE Trans. Pattern Anal. Mach. Intell. 35(4), 849–862 (2013)

[6] Gu, S., Zhang, L., Zuo, W., Feng, X.: Weighted nuclear norm minimization with application to image denoising. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2862–2869 (2014)

[7] Schmidt, U., Roth, S.: Shrinkage fields for effective image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2774–2781 (2014)

[8] Chen, Y., Pock, T.: Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration. IEEE Trans. Pattern Anal. Mach. Intell. 39(6), 1256–1272 (2017)

[9] S. Rahman, M. M. Rahman, M. Abdullah-Al-Wadud, G. D. Al-Quaderi, and M. Shoyaib, “An adaptive gamma correction for image enhancement,” EURASIP Journal on Image and Video Processing, vol. 2016, no. 1, pp. 1–13, 2016

[10] H. D. Cheng and X. J. Shi, “A simple and effective histogram equalization approach to image enhancement,” Digital Signal Processing, vol. 14, no. 2, pp. 158–170, 2004.

[11]S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz,T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, “Adaptive histogram equalization and its variations,” Computer vision, graphics, and image processing, vol. 39, no. 3, pp. 355–368, 1987.

[12] S. M. Pizer, “Contrast-limited adaptive histogram equalization: Speed and effectiveness stephen m. pizer, r. eugene johnston, james p. ericksen, bonnie c. yankaskas, keith e. muller medical image display research group,” in Proceedings of the First Conference on Visualization in Biomedical Computing, Atlanta, Georgia, vol. 337, 1990.

[13] L. Wang, L. Xiao, H. Liu, and Z. Wei, “Variational bayesian method for retinex,” IEEE Transactions on Image Processing, vol. 23, no. 8, pp.3381–3396, 2014.

[14] W. Y. Chen Wei, Wenjing Wang and J. Liu, “Deep retinex decomposition for low-light enhancement,” in British Machine Vision Conference. British Machine Vision Association, 2018.

[15] X. Dong, G. Wang, Y. Pang, W. Li, J. Wen, W. Meng, and Y. Lu, “Fast efficient algorithm for enhancement of low lighting video,” in 2011 IEEE International Conference on Multimedia and Expo. IEEE, 2011, pp.1–6.

[16] L. Li, R. Wang, W. Wang, and W. Gao, “A low-light image enhancement method for both denoising and contrast enlarging,” in 2015 IEEE International Conference on Image Processing (ICIP). IEEE, 2015, pp. 3730–3734.

[17] Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. IEEE Trans. Image Process. 26(7), 3142–3155 (2017)

[18] S. Ioffe and C. Szegedy, ``Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proc. Int. Conf. Mach. Learn., Lille, France, 2015, pp. 448-456.

[19] Y. Tai, J.Yang, X. Liu, and C. Xu, ``MemNet :A persistent memory network for image restoration," in Proc. IEEE Int. Conf. Computer. Vision. (ICCV), Venice, Italy, Oct. 2017, pp. 4549-4557.

[20] K. Zhang, W. Zuo, and L. Zhang, ``FFDNet: Toward a fast and flexible solution for CNN-based image denoising," IEEE Trans. Image Process., vol. 27, no. 9, pp. 4608-4622, Sep. 2018.

[21] Lan, R., Zou, H., Pang, C. et al. Image denoising via deep residual convolutional neural networks. SIViP 15, 1–8 (2021). https://doi.org/10.1007/s11760-019-01537-x

[22] He, K., Zhang, X., Ren, S., Sun, J.: Identity Mappings in Deep Residual Networks. Springer, Berlin (2016)

[23] O. Ranneberger, P. Fischer, and T. Brox, ``U-net: Convolutional networks for biomedical image segmentation," in Proc. Int. Conf. Med. Image Computer Assist. Intervent. Cham, Switzerland: Springer, 2015, pp. 234-241.

[24] K. G. Lore, A. Akintayo, and S. Sarkar, “Llnet: A deep autoencoder approach to natural low-light image enhancement,” Pattern Recognition, vol. 61, pp. 650–662, 2017.

[25] W. Y. Chen Wei, Wenjing Wang and J. Liu, “Deep retinex decomposition for low-light enhancement,” in British Machine Vision Conference. British Machine Vision Association, 2018.

[26] C. Guo, C. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, and R. Cong, “Zero reference deep curve estimation for low-light image enhancement,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1780–1789.

[27] Y. Jiang, X. Gong, D. Liu, Y. Cheng, C. Fang, X. Shen, J. Yang, P. Zhou, and Z. Wang, “EnlightenGAN: Deep light enhancement without paired supervision,” IEEE Transactions on Image Processing, vol. 30, pp. 2340–2349, 2021.

[28] Y. Zhang, J. Zhang, and X. Guo, “Kindling the darkness: A practical low-light image enhancer,” in Proceedings of the 27th ACM international conference on multimedia, 2019, pp. 1632–1640.

[29] Wei C, Wang W, Yang W, Liu J. Deep Retinex decomposition for low-light enhancement. arXiv preprint arXiv:1808.04560. 2018 Aug 14.

[30] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[31] Brooks T, Mildenhall B, Xue T, Chen J, Sharlet D, Barron JT. Unprocessing images for learned raw denoising. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019 (pp. 11036-11045).

[32] Xiong W, Liu D, Shen X, Fang C, Luo J. Unsupervised low-light image enhancement with decoupled networks. In2022 26th International Conference on Pattern Recognition (ICPR) 2022 Aug 21 (pp. 457-463). IEEE.

[33] F. Chollet et al., “Keras,” https://github.com/fchollet/keras, 2015.

[34] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” CoRR, 2015

[35] V. Jain and S. Seung, “Natural image denoising with convolutional networks,” in Advances in Neural Information Processing Systems, 2009, pp. 769–776.

[36] X. Lan, S. Roth, D. Hutten ocher, and M. J. Black, “Efficient belief propagation with learned higher-order Markov random fields,” in European Conference on Computer Vision, 2006, pp. 269–282.

[37] L. Jiang et al., “Deep refinement network for natural low-light image enhancement in symmetric pathways,” Symmetry 10(10), 491 (2018).

[38] Lin G, Milan A, Shen C, Reid I. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 1925-1934).

[39] J. Wang et al., “RDGAN: retinex decomposition based adversarial learning for low-light

enhancement,” in IEEE Int. Conf. Multimedia and Expo, IEEE, pp. 1186–1191 (2019).

[40] M. Zhu et al., “EEMEFN: low-light image enhancement via edge-enhanced multi-exposure

fusion network,” in Proc. AAAI Conf. Artif. Intell., Vol. 34, No. 7, pp. 13106–13113

(2020).

[41] A. Zhu et al., “Zero-shot restoration of underexposed images via robust retinex decomposition,”in IEEE Int. Conf. Multimedia and Expo, IEEE, pp. 1–6 (2020).

[42] Zamir SW, Arora A, Khan S, Hayat M, Khan FS, Yang MH, Shao L. Learning enriched features for real image restoration and enhancement. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020.

[43] Liu, X., Zhang, Q., Hou, Y., Liu, C., & Wang, X. (2019). Dual Illumination Enhancement Network for Low-Light Image Enhancement. In Proceedings of the IEEE International Conference on Computer Vision (pp. 4647-4656).

[44] Tom Mertens, Jan Kautz, and Frank Van Reeth. Exposure fusion. In PCCGA, 2007.

[45] Tom Mertens, Jan Kautz, and Frank Van Reeth. Exposure fusion: A simple and practical alternative to high dynamic range photography. Computer Graphics Forum, 28(1):161–171, 2009.

[46] Gershon Buchsbaum. A spatial processor model for object colour perception. J. Franklin Institute, 310(1):1–26, 1980.

[47] Charbonnier, P., Blanc-Feraud, L., Aubert, G., Barlaud, M.: Two deterministic half-quadratic regularization algorithms for computed imaging. In: ICIP (1994)

[48] Szeliski R. Computer vision: algorithms and applications. Springer Nature; 2022 Jan 3.

[49] Tripathi M. Facial image denoising using Autoencoder and UNET. Heritage and Sustainable Development. 2021;3(2):89.

[50] Wikipedia contributors. (2023, March 19). Bayer filter. Wikipedia. https://en.wikipedia.org/wiki/Bayer_filter

[51] Google. (n.d.). Normalization. Retrieved April 4, 2023, from https://developers.google.com/machine-learning/data-prep/transform/normalization

[52] Kim, S. E., & Seo, I. W. (2015). Artificial Neural Network ensemble modelling with conjunctive data clustering for water quality prediction in rivers. Journal of Hydro-Environment Research, 9(3), 325-339.

[53] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016, pp. 770–778.

[54] O. Ranneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in MICCAI, 2015, pp. 234–241.

[55] Sevastopol sky, A., Drapak, S., Kiselev, K., Snyder, B. M., Keenan, J. D., & Gordievsky, A. (2018). Stack‐u‐net: refinement network for image segmentation on the example of optic disc and cup. arXiv preprint arXiv:1804.11294.

[56] Wang, Zhou; Bovik, A.C.; Sheikh, H.R.; Simonelli, E.P. (2004-04-01). "Image quality assessment: from error visibility to structural similarity". IEEE Transactions on Image Processing. 13 (4): 600–612.

[57] Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14 (pp. 694-711). Springer International Publishing.

[58] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR), 2015

[59] Zamir, S. W., Arora, A., Khan, S., Hayat, M., Khan, F. S., Yang, M. H., & Shao, L. (2020). Learning enriched features for real image restoration and enhancement. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16 (pp. 492-511). Springer International Publishing.

[60] Guo, C., Li, C., Guo, J., Loy, C. C., Hou, J., Kwong, S., & Cong, R. (2020). Zero‐reference deep curve estimation for low-light image enhancement. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 1780-1789).

[61] Wikipedia contributors. (2022, June 29). Histogram equalization. Wikipedia. https://en.wikipedia.org/wiki/Histogram_equalization

[62] Mu, W., Liu, H., Chen, W., & Wang, Y. (2022c, September 1). A More Effective Zero-DCE Variant: Zero-DCE Tiny. Electronics; MDPI. https://doi.org/10.3390/electronics11172750