VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAUM-590 014

A Project Report

on

"LOW LIGHT IMAGE ENHANCEMENT USING DEEP LEARNING"

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

CHARISH PATEL M N	**1VE18CS041**
CHUNITH GOWDA G S	**1VE18CS043**
RASHMI S	**1VE18CS134**
SHIRISHA K S	**1VE18CS146**

Under the Guidance of

Prof. Balaji K

Assistant Professor

Department of Computer Science and Engineering

Sri Venkateshwara College of Engineering, Bengaluru-562 157

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESHWARA COLLEGE OF ENGINEERING

BANGALORE - 562 157.

2021-2022

Accredited by NAAC & NBA*

SVCE BENGALURU

SRI VENKATESHWARA COLLEGE OF ENGINEERING

— Affiliated to VTU, Approved by AICTE, Recognised by UGC u/s 2(f) & 12(B)—

SRI VENKATESHWARA COLLEGE OF ENGINEERING
Vidyanagar, Bangalore – 562 157

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled **"LOW LIGHT IMAGE ENHANCEMENT USING DEEP LEARNING"** carried out by **Mr. Charish Patel M N (1VE18CS041) , Mr. Chunith Gowda G S (1VE18CS043), Ms. Shirisha K S (1VE18CS146) and Ms. Rashmi S (1VE10CS134)** a bonafide student of Sri Venkateshwara College of Engineering, in partial fulfillment of Bachelor of Engineering in Computer Science and Engineering of **Visvesvaraya Technological University, Belgaum** during the academic year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Signature of the Guide	**Signature of the HOD**	**Signature of the Principal**
Prof. Balaji K	**Dr. S. C. Lingareddy**	**Dr. Nageshwara Guptha M**
Assistant Professor, SVCE	HOD, SVCE	Principal, SVCE
Bangalore	Bangalore	Bangalore

Name of the examiners:	**Signature with date**
1.	
2.	

ABSTRACT

During the past years, deep convolutional neural networks have achieved impressive success in low-light Image Enhancement. Existing deep learning methods mostly enhance the ability of feature extraction by stacking network structures and deepening the depth of the network. In order to reduce inference time while fully extracting local features and global features. Inspired by SGN ,we propose an Attention based Broadly Self-guided Network (ABSGN) for real world low-light image Enhancement. The basic structure of ABSGN is a top-down self-guidance architecture which is able to efficiently incorporate multi-scale information and extract good local features to recover clean images. Moreover, such a structure requires a smaller number of parameters and enables us to achieve better effectiveness than UNet structure. In addition, the proposed network comprises several multi-level guided Dense Blocks (MGDB) which can be viewed as a novel extension of Dense block in feature space. At the lowest resolution level of ABSGN, we offer more efficient module to fully extract the global information to generate the better final output, which called Global Spatial Attention (GSA). Such broad strategy is able to handle the noise at different exposures. The proposed network is validated by many mainstream benchmarks. Additional experimental results show that the proposed network outperforms most of state-of-the-art low-light image enhancement solutions.

ACKNOWLEDGEMENT

At our very outset, we would like to place on record our deepest gratefulness towards the Visvesvaraya Technological University for including the task of doing this project in our syllabus, thus helping us to develop immense interest and a practical approach.

I am grateful to my institution, **SRI VENKATESHWARA COLLEGE OF ENGINEERING** with its ideals and inspiration for having provided us with the facilities, which has made this project work a success.

We express our sincere gratitude to **Dr. Nageswara Guptha M, Principal, SVCE** for giving the opportunity to embark upon this project.

We express our profound gratitude to **Dr. S. C. Lingareddy, HOD, Dept. of CSE, SVCE** and the whole department for providing us kindly environment for the successful completion of the project work.

We also extend my sincere thanks to my project guide **Mr. Balaji K, Asst Prof, Dept of CSE, SVCE** and project coordinator **Mrs. Kulkarni Varsha, Asst Prof, Dept of CSE, SVCE** for the timely suggestions and cooperation throughout the execution of the project.

We would like to thank teaching and non-teaching staff of Dept of CSE for their support.

We would like to thank our beloved parents and friends for their immense and moral support throughout our journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER -01

INTRODUCTION

Low contrast, poor visibility, and noise characterizes images captured in low-light circumstances. These limitations provide a difficulty to both human vision, which favors high-visibility images, and many intelligent systems that rely on computer vision algorithms, such as all-day autonomous driving and biometric recognition. A significant number of techniques have been proposed to alleviate the degradation, ranging from histogram or cognition-based approaches to learning-based approaches. Deep learning approaches to image restoration and improvement, such as super-resolution, denoising, and deblurring, significantly rely on either synthetic or recorded corrupted and clean image pairs to train.

Despite the fact that numerous methods have been developed for low light image enhancement in recent years, there is still much space for improvement. Histogram equalization-based approaches seek to boost contrast by merely expanding the dynamic range of images, whereas Retinex-based methods use the predicted illumination map to recover contrast. They mostly concentrate on restoring brightness and contrast while ignoring the effects of noise. In reality, noise is unavoidable and noticeable in low-light photographs, especially when brightness and contrast are increased. Low-light Image Enhancement is a fundamental task in low-level vision and an important pre-processing step in many other vision tasks. images captured under the unsuitable lighting are either too dark or too bright. The art of recovering the original clean image from its corrupted measurements is studied under the image restoration task. It is an ill-posed inverse problem, due to the existence of many possible solutions. While we enhancing the brightness, we also need to tackle the color distortion, the amplified noise, the loss of detail and texture information and the blurred edges.

1.1 Project Domain

The project mainly focuses on low light image enhancement and as the name suggests the domain of this project would be where the user needs an enhanced image. This could be a scenario where the user needs to pre-process the image before using it further. Considering cases like CCTV cameras installed for security purposes take images at low light conditions. If these images were to be enhanced, the system would be useful. Basically, any system that takes low light images as input can have this enhancing feature before processing the images.

1.2 Issues and Challenges

The major issue for the project would be the collection of datasets and training the model. Datasets of images under various lighting conditions are required for training the model. Images under low lighting conditions and corresponding images under good lighting conditions are to be collected and annotated. They have to be segregated for testing, training and validation.

1.3 Need for ML Based Solutions

Image enhancement can be very well done by ML. The enhancement of images includes changing various features of pixels of an image like brightness, saturation, contrast etc. This can be efficiently done using deep learning techniques like CNN. This will assure minimum loss and maximum quality output.

1.4 Problem Statement

To enhance the illumination of low light images captured under various lighting , Enhancement of low light images might be required at times of processing them later. This can be done using various deep learning techniques. One among them is CNN which helps in enhancing the images and giving them a better aesthetic look and would be of greater quality.

1.5 Objectives

- To illuminate the images captured in low light conditions.
- Minimize the image quality degradation after enhancement.
- Coping up with diverse lighting conditions such as non-uniform and poor lighting conditions.
- To brighten up the image while preserving the inherent color and details.

CHAPTER- 02

LITERATURE SURVEY

[1] A REVIEW ON IMAGE ENHANCEMENT WITH DEEP LEARNING APPROACH (2018).

This paper aimed at enhancing the images using Deep neural networks. This method of enhancing the images is a lot time taking and includes numerous calculations for decreasing noise levels.

There are many methods for Image Enhancement some of them are listed below:

a) Histogram matching: Histogram matching is the change of an image with the goal that its histogram coordinates a predefined histogram. The surely understood histogram equalization strategy is an exceptional case in which the predetermined histogram is consistently appropriated.

b) Contrast-limited adaptive histogram equalization (CLAHE): It is used to enhance the contrast of the grayscale image assumed by transforming technique. CLAHE works on small regions in the image called tiles, rather than the whole image. Contrast of every tile is enhanced; therefore, the histogram of the output region approximately matches the histogram predefined by the “Distribution” parameter.

c) Wiener filter: Wiener filter is a filter used to create a gauge of a coveted or target arbitrary process by linear time-invariant (LTI) filtering of an observed noisy process, accepting known stationary signal and noise spectra, and added substance noise. The Wiener filter limits the mean square error between the evaluated random process and the desired procedure.

d) Median filter: The median filter is a nonlinear computerized filtering method, regularly used to expel noise from an image. Such noise reduction is a common pre-processing step to enhance the results of later processing for example, edge recognition on an image. Median filtering is broadly utilized as a part of digital image processing in light of the fact that, under specific conditions, it preserves edges while removing noise.

e) Linear contrast adjustment: In this the contrast adjustment block changes the contrast of an image by linearly scaling the pixel values amongst lower and upper limits. Pixel values that are below or above this range are saturated to the lower or upper limit value, individually.

f) Unsharp mask filtering: Unsharp masking (USM) is an image sharpening method, frequently accessible in digital image processing software. The "unsharp" of the name comes from the way that the procedure utilizes an obscured, or "unsharp", negative image to make a mask of the original image. The unsharp mask is then joined with the positive (original) image, constructing an image that is less blurred than the original. The subsequent image, in spite of the fact that it is clearer, might be a less precise portrayal of the image's subject.

g) Deep neural network: Execute image processing undertakings, for example, removing noise from images and constructing high-resolution images from low-resolutions images, utilizing convolutional neural networks. Deep learning utilizes neural networks to learn valuable portrayals of highlights straightforwardly frominformation. For instance, you can utilize a neural network to recognize the images and remove various type ofnoise from images.

[2] AUTOMATIC IMAGE QUALITY ENHANCEMENT USING DEEP NEURAL NETWORKS (2019).

This research proposes a novel dataset generation method for automated image enhancement research, and tests its usefulness with the chosen network design. This dataset generation method simulates commonly occurring photographic errors, and the original high-quality images can be used as the target data. This dataset design allows studying fixes for individual and combined aberrations. Limitation of this is the method only analyses the pixel values and image statistics without any information about context or semantics, the algorithm may fail to apply the enhancements in such a way that a human observer finds pleasing. Color transformations can be applied as a countermeasure for the errors. More detail is provided on these photographic flaws and previously developed solutions are described alongside some demonstrations of the effects of these enhancements. It should be noted that even though this section is categorized as different operations, these flaws are not separate from each other and the enhancement methods often overlap. Many of the image enhancements are applied in the RGB color space, which is most often used to represent the pixel color values. In addition, the HSV saturation, and value) color space can be used. HSV is designed to represent individual aspects of human color perception instead of separate color channels as in RGB.

[3] LOW CONTRAST IMAGE ENHANCEMENT USING CONVOLUTIONAL NEURAL NETWORKWITH SIMPLE REFLECTION MODEL (2019).

The proposed method consists of four steps in total and first step is the low contrast estimation step that generates a low contrast probability map. The second step is a create contrast gray scale step. The third step is a refining probability map step with obtained probability map and converted gray scale image. The final step is an enhancement step that enhances low contrast image with refined probability map and converted gray scale.

[4] DEEP LEARNING FOR IMAGE ENHANCEMENT AND VISIBILITY IMPROVEMENT (2019).

This paper attempts to apply deep learning to image filtering, specifically low-light image enhancement. Work will then be done to produce a fully functioning image filtering system using deep learning, which will allow the network to be trained using supervised learning, and filtered output images to be saved to a file. Overall, the network was successful, producing output images which can clearly be seen to be filtering low-light images. The network will need to be run for a greater amount of time to see the best possible results the network can output.

[5] GETTING TO KNOW LOW-LIGHT IMAGES WITH THE EXCLUSIVELY DARK DATASET (2018).

This paper presents two contributions. First, is the Exclusively Dark (ExDARK) dataset which is the largest collection of low-light images taken in visible light to-date with object level annotation. Secondly, it provides an object-focused analysis of low-light images using the state-of-the-art algorithms in both hand-crafted and learned features for a better understanding of low-light vision and its difference from vision with sufficient illumination. Types of low-light.

• Low: Images with very low illumination and hardly visible details.

• Ambient: Images with weak illumination and the light source is not captured within.

• Single: Images where a single light source is visible.

• Weak: Images with multiple visible but weak light sources.

• Strong: Images with multiple visible and relatively bright light sources.

[6] QUANTIFYING COLOR CONSTANCY: EVIDENCE FOR NONLINEAR PROCESSING OF CONE- SPECIFIC CONTRAST (2019).

Color constancy was studied by the method of comparing color samples under two different illuminants using a CRT color monitor. In addition to the classical approach in which one of the illuminants is a (standard) white, we performed experiments in which the range of differential illumination was extended by using pairs of lights that were both colored. The stimulus pattern consisted of an array of thirty- five color samples (including five neutral samples) on a white background. A trichromatic illuminant-object interaction was simulated analogous to that resulting from illumination by three monochromatic lights. The test samples, as seen under “test” and “match” illumination, were successively presented to the left and right eye (haploscopic matching). The data show systematic deviations from predictions on the basis of cone-specific normalization procedures like those incorporated in the retinex algorithm and the von Kries transformation. The results can be described by a nonlinear response transformation that depends on two factors, receptor- specific sample/background contrast and the extent to which the illuminant stimulates the receptor system in question. The latter factor explains the deviations. These are mainly caused by the short-wave-sensitive system, as a consequence of the fact that this system can be more selectively stimulated than the other, spectrally less separated, cone systems.

CHAPTER-03
HARDWARE AND SOFTWARE REQUIREMENTS

3.1 Software requirements

• Operating System: Windows 10 or higher

• Language: Python 3.8 or above

• IDE/Tool: Visual studio, Command Prompt

• Python libraries: PyTorch, OpenCV, Pillow, matplotlib, Pillow

3.2 Hardware requirements

• CPU: Intel Core i3 and above

• GPU: Quadro RTX 5000 (8GB VRAM) or equivalent

• Processor: 4 Cores at 2.8 Hz

• RAM: 4GB or higher

• Disk Space: 15GB.

• Monitor: Preferably color monitor (16-bit color) and above.

CHAPTER – 04

SYSTEM ANALYSIS

4.1 Architecture

In this section, we firstly introduce the overall network structure and then introduce the details of ABSGN

4.1.1 Overall Structure of ABSGN

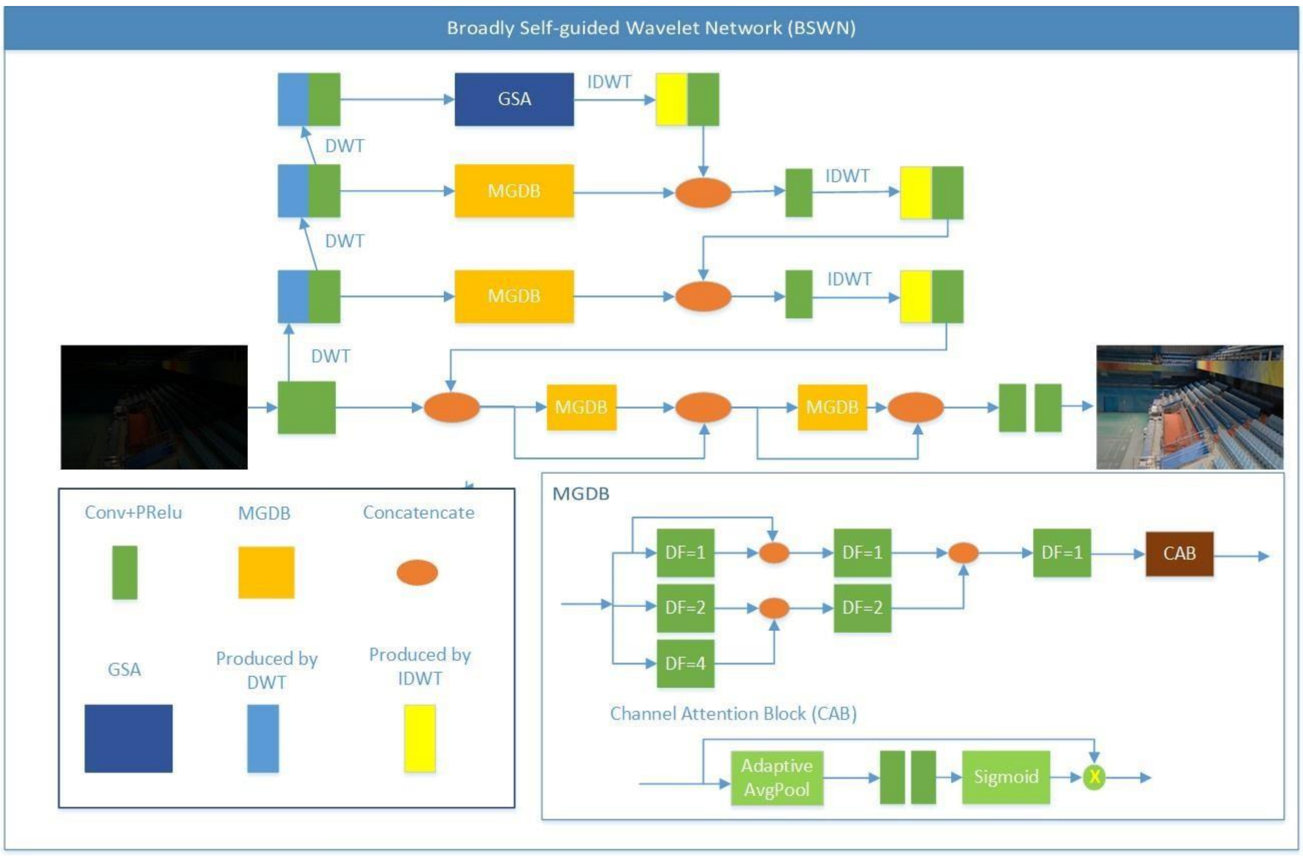

Fig 4.1: Architecture of ABSGN

Our proposed network uses a top-down self-guidance architecture to better exploit image multi-scale information. Information extracted at low resolution is gradually propagated into the higher resolution sub-networks to guide the feature extraction processes. Firstly, we pass the input image through a Conv+PReLU layers to obtain a main feature map with 32 channels. Then instead of Pixel Shuffle and Pixel UnShuffle, DWT and IDWT are used to generate multiscale inputs. ABSGN uses wavelet transform to transform the main feature map to three smaller scales. At the lowest resolution layer, we offer a Global Spatial Attention (GSA) Block including spatial attention module. At the full resolution layer, we adopt Dense connections consists of two MGDB to improve the reuse of the main feature map. In the rest of this paper, we simply refer to the advantages of GSA and MGDB. we also make a comparative experiment to prove our conclusion. In addition, we find batch normalization is harmful for the denoising performance and only use one normalization layer in this network.

4.1.2 Detail Structure of ABSGN

Different form DSWN, we firstly acquire the main feature map with 32 channels from the input image by a Conv+PReLU layer. We use DWT and a Conv+PReLU layer to finish the down sampling process. Performing the above down sampling operation three times in sequence to obtain feature maps of different sizes.

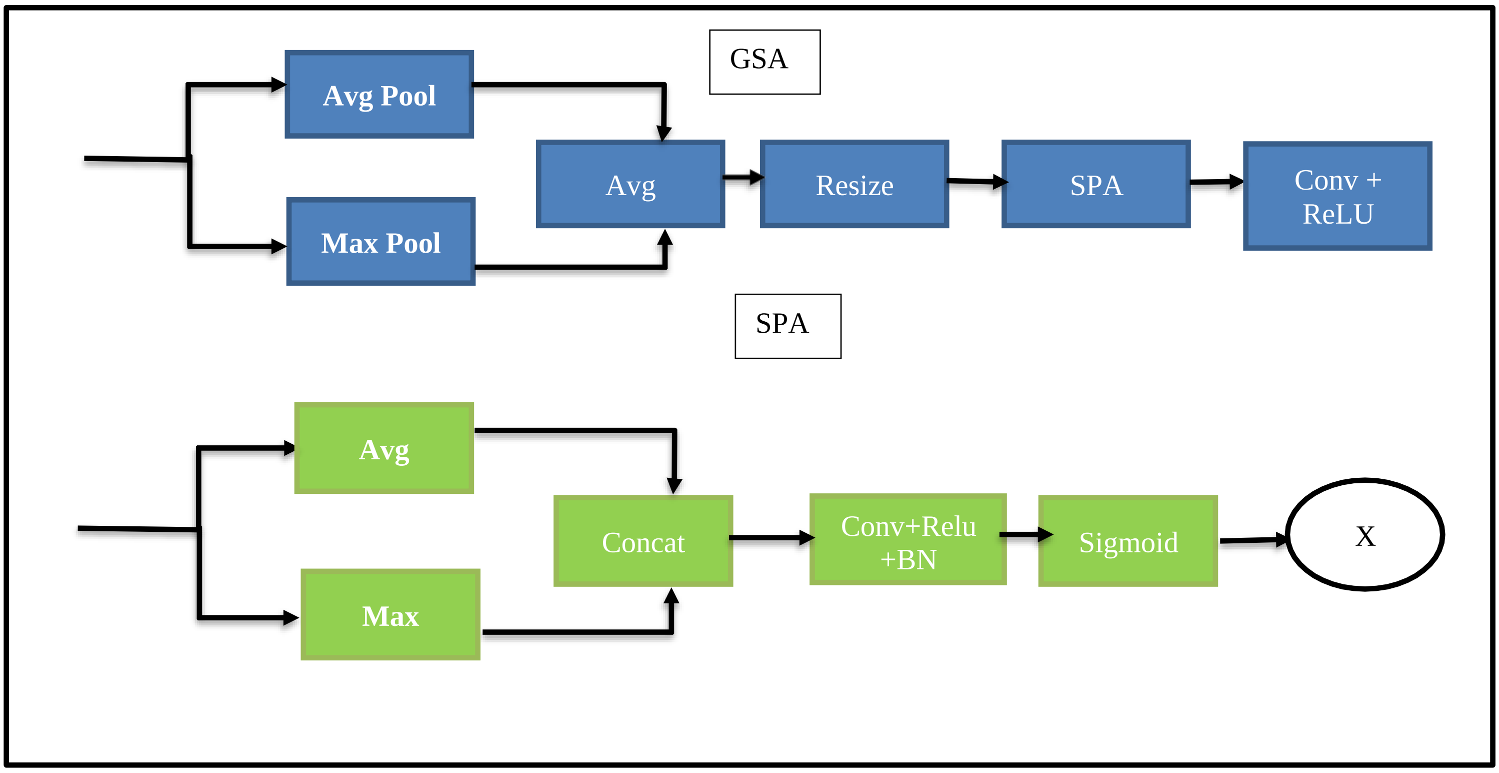

Fig 4.2: Detailed ABSGN View

The top level of ABSGN works on the smallest spatial resolution to extract large scale information. As is shown in Figure, the top sub-network uses the GSA block to get the global information. which contains two

Conv+PReLU layers and AdaptiveAvgPool2d, AdaptiveMaxPool2d, interpolation (the Resize block in Figure) and Spatial Attention module (SPA). Particularly, given an input feature map, i.e. X, with a size of H × W × C, AdaptiveAvgPool2d and AdaptiveMaxPool2d is employed to extract the representative information. Average of them is used for the global information producing a feature map with a size of 1 × 1 × C. Then, an interpolation function is utilized to upscale the feature map with global information which is processed by a Conv+PReLU to shrink the number of channels, yielding a global feature map with a size of H × W × C1. Then we apply SPA block to increase attention to different areas of the global feature map.

As for the full resolution level, we add more MGDB to reuse the main feature map, which enhance the feature extraction capability of ABSGN, after merging information from all the scales, we use two Conv+PReLU layers to acquire the final output as the restored image. By adding gradient loss, our network is able to achieve better retention of details without reducing PSNR. Inspired by a new joint loss function, our network uses L1 loss plus SSIM loss for training. The total loss is as follows:

$$\mathbf{L_{ABSGN} = \gamma L_{SSIM}(I\text{-}i)+(1\text{-}\gamma\ )L_1(I\text{-}i)}$$

where $\gamma \in [0, 1]$ is the weight to balance the two terms. Here we choose the value of the γ is 0.16.

4.2 Incorporation of multi-scale information

To extract multi-scale information for image restoration tasks, Pixel Shuffle and wavelet transform have been proposed to replace pooling and interpolation to avoid information loss. With self-guidance strategy and Pixel Shuffle, SGN greatly improved the memory and runtime efficiency. Bae et.al proposed a wavelet residual network (WavResNet) for image denoising and SISR and find wavelet sub bands benefits learning convolutional neural network (CNN). Multi-level wavelet transform is considered by DSWN to achieve better receptive field size and avoid down-sampling information loss by embedding wavelet transform into CNN architecture. DSWN owns more power to model both spatial context and inter-sub band dependency by embedding DWT and IDWT to CNN. In this paper, our proposed network adopts the same method as DSWN to incorporate multi-scale information with a totally different architecture from DSWN.

4.3 Feature Extraction

As we know, stacking multiple convolutional layers can help us effectively extract high-dimensional information in the feature map. However, this will lead to a substantial increase in network parameters, requiring the massive amount of training data to prevent overfitting. The larger the size of the convolution kernel used, the larger the receptive field. But it will bring heavy computation and slower running time. With Densely Connected Residual Block, DRNet mitigates the problems of overfitting, vanishing gradient, and training instability during training very deep and wide networks.

4.4 Methodology

The proposed system makes use of ABSGN. It is a class of deep neural networks applied to analyse visual imagery. These contain various layers which generate various activation functions which are given as input to the next layer. Using this technique, we aim at manipulating the pixels of the low light image to change their attributes and thus enhancing the image.

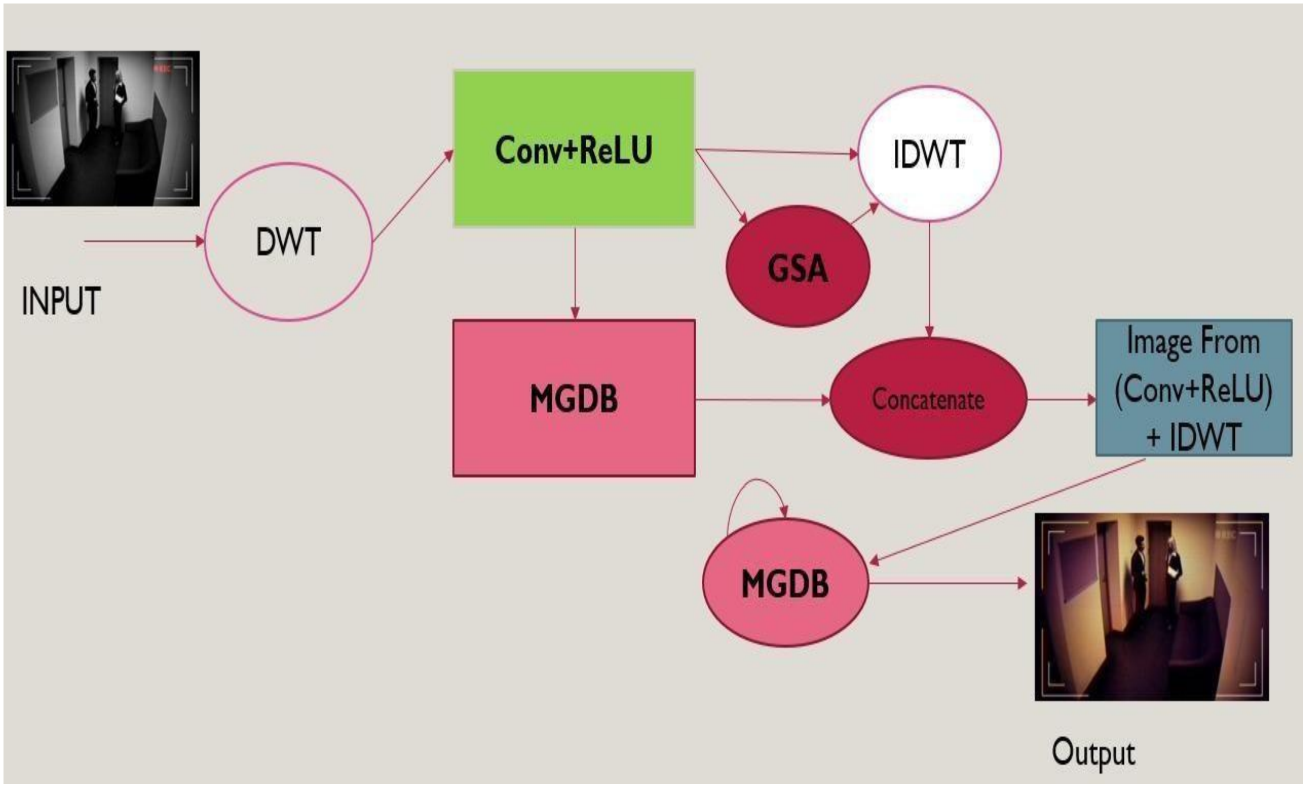

Fig 4.3: Methodology

The important components of the network are as follows.

4.4.1 Convolution layer

The whole network has 7 convolution layers. Each layer consists of multiple kernels, and the weights of these kernels do not change during the convolution process, i.e., there is weight sharing. With the convolution operation, it extracts the different features of the input images at different convolution levels. The output of the first CNN layer roughly depicts the location of low-level features (edges and curves) in the original image. On this basis, another convolution operation is carried out, and the output will be the activation map representing higher-level features. Such features can be semicircles i.e., a combination of curves and lines or quadrilateral i.e., a combination of several lines. The more convolution layers, the more complex feature activation map will be obtained. There are several parameters that need to be determined for each layer, such as the kernel size K, padding P, and stride S. The number of kernels N is the number of output feature maps.

4.4.2 Activation Function

The activation function is vital in a deep CNN because the nonlinearity of the activation layer introduces nonlinear characteristics to a system which has just undergone linear computation. We have adopted a rectified linear unit (ReLU) for its advancement in improving the training speed without obvious changes in accuracy. The activation function is applied over the output of the previous layer. Every value obtained from the upper stream convolution layer should be activated by ReLU before it is input into the downstream convolution layer.

4.4.3 Max-pooling operation

The pooling operation is helpful to reduce the number of parameters and to resize the image to the original patch image size, decreasing the training cost by a meaningful extent. The pooling operation can also cut down the possibility of overfitting, helpful to suppress noise. For the model, we have set the kernel size to 2 for each max-pooling operation.

4.5 Dataset Details

Low-Light (LOL) dataset is a publicly available dark-light paired images dataset in the real sense. The lowlight images are collected by changing exposure time and ISO. It contains 500 images in total, The resolution of each of these images is 400 × 600. we use 485 images of them for training, and the rest for evaluation as suggested by Most state-of-the-art low-light image enhancement solutions select the LOL as their training dataset. To train our ABSGN model, we also use LOL dataset as training and validation dataset for low-light image enhancement task. The LOL dataset consists of two parts: the real-world dataset and the synthetic dataset. Because the LOL synthetic dataset cannot simulate the degradation of real-world images very well, we deprecate the synthetic dataset where low-light images are synthesized from normal-light images and only use the real-world dataset.

CHAPTER - 05

SYSTEM DESIGN

5.1 Workflow

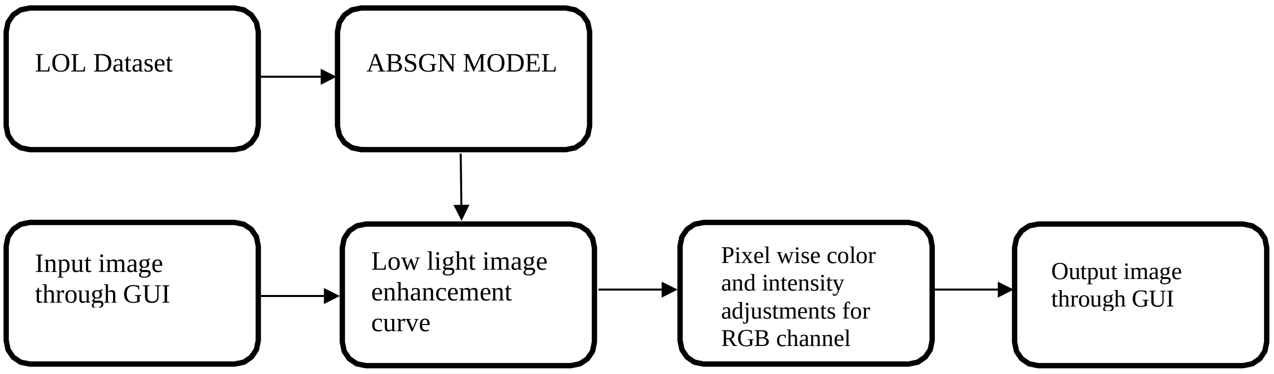
LOL Dataset
ABSGN MODEL
Input image through GUI
Low light image enhancement curve
Pixel wise color and intensity adjustments for RGB channel
Output image through GUI

Fig 5.1: Workflow Diagram

ABSGN model is trained with the dataset and saved for further use. In the process, loading of the pre-trained model is the first step. Low light image is given as input to the network. The corresponding illuminated image that is obtained is displayed in GUI and can be saved in any local directory In Addition, we try to apply the custom convolution to take place of all the dilation convolutions, which will tell the necessity of introducing dilation convolution. The results suggest that MGDB can get more local information and better performance in SSIM/PSNR. Similarity, we respectively use SPA module and GIA module substitute for our GSA module to prove the advantage of our proposed module. Compared to SPA module and GIA module, our module has irreplaceable advantage.

5.2 Use cases

A use case is a description of the ways in which a user interacts with a system or product. A use case may establish the success scenarios, the failure scenarios, and any critical variations or exceptions. A use case can be written or made visual with the help of a use case model tool. When presented in written form, a use case can be a helpful piece of project documentation. Use cases are a common requirements artifact and they can smoothen communication across technical and business stakeholders.

A use case document should establish and identify a few key components — these are:

1. System: A system is the product, service, or software under discussion.

2. Actors: An actor is a user or anything else that exhibits behavior when interacting with the system. The actor could be another system, a piece of hardware, or an entire organization.

3. Scenario: A scenario is a specific sequence of actions and interactions between actors and the system under discussion; it is also called a use case instance.

4. Use cases: A use case outlines the success and failure scenarios that can occur when the actor(s) interact with the system.

5.3 Use case for image enhancement application

Users can use the GUI to enhance the image. When the user opens the GUI, they are given the option to upload any image to enhance by clicking the UPLOAD IMAGE button. The user can select any image of JPG format from the appearing dialog box. Once the image is selected, it appears on the GUI. If the selected image is to be changed, the user can upload a new image by clicking on the same button. The next step will be to enhance the image by clicking on the ENHANCE IMAGE button. The image is enhanced by the model and the result is displayed on the GUI. The user can save the image by clicking on the SAVE IMAGE button which will prompt a dialog box to the user. The user can navigate to any directory and give the required name for the image. The image by default is saved as JPG in the specified directory.

System: Image enhancement application

Primary actor: Any person using the application

Scenario: The user runs the GUI application. Uploads an image of his choice. This image is sent to the model which is pre trained and helps in enhancement of the image. The model after processing gives the output image which is displayed on the GUI. If the user is not satisfied with the output, he can re enhance the image. Once done, the user also has the option to save the output image to any folder in the system by any name with default JPG extension. This can be repeated for any number of images.

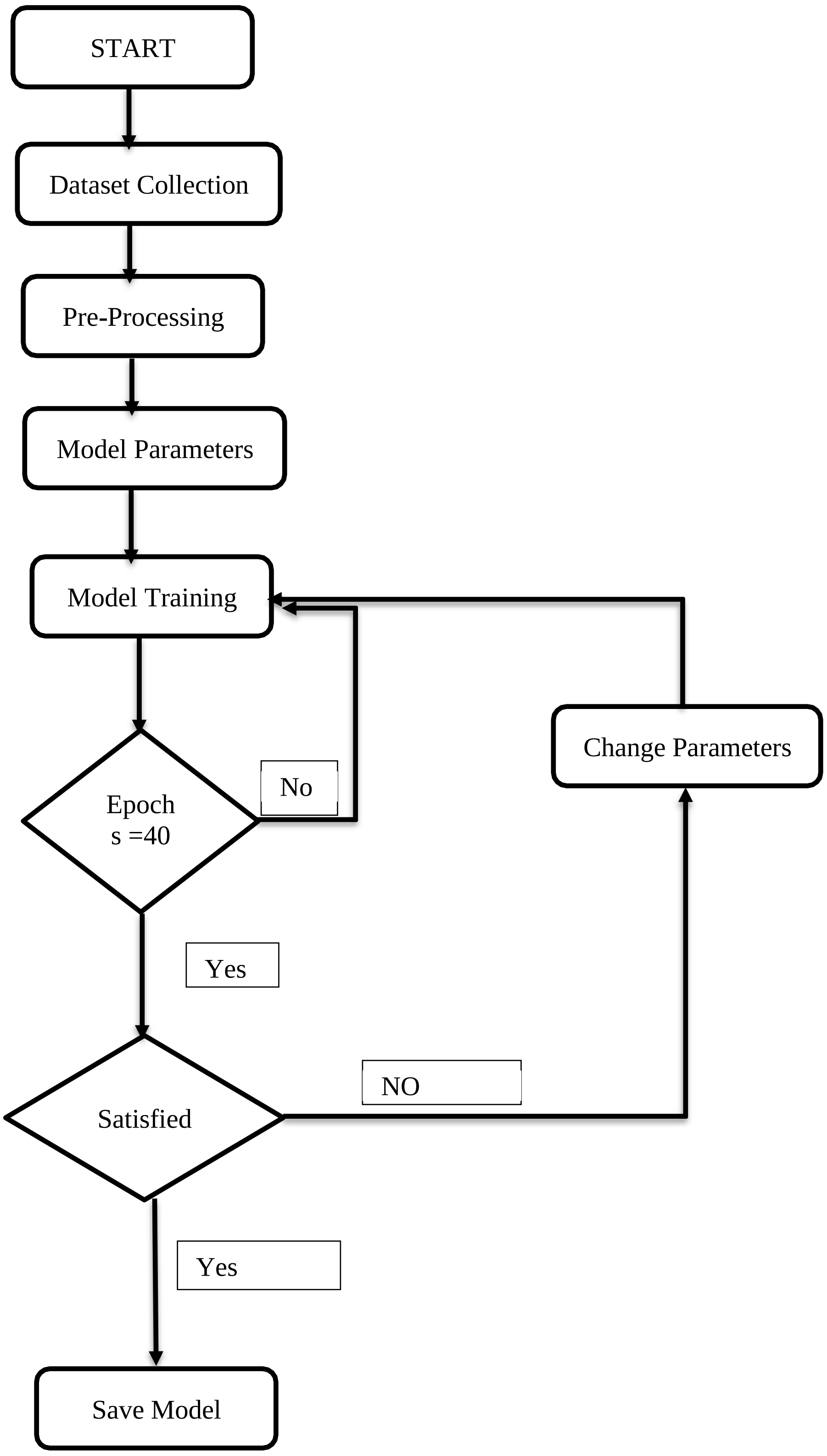

Fig 5.2: Flow diagram of Model data processing

The above diagram represents the sequence of activities involved starting from the dataset collection, dataset pre-processing and deciding model parameters. Once this is done the model is trained with 40 epochs. Further if the model performs well, it is saved. Else the model parameters are changed and training of the model is done again.

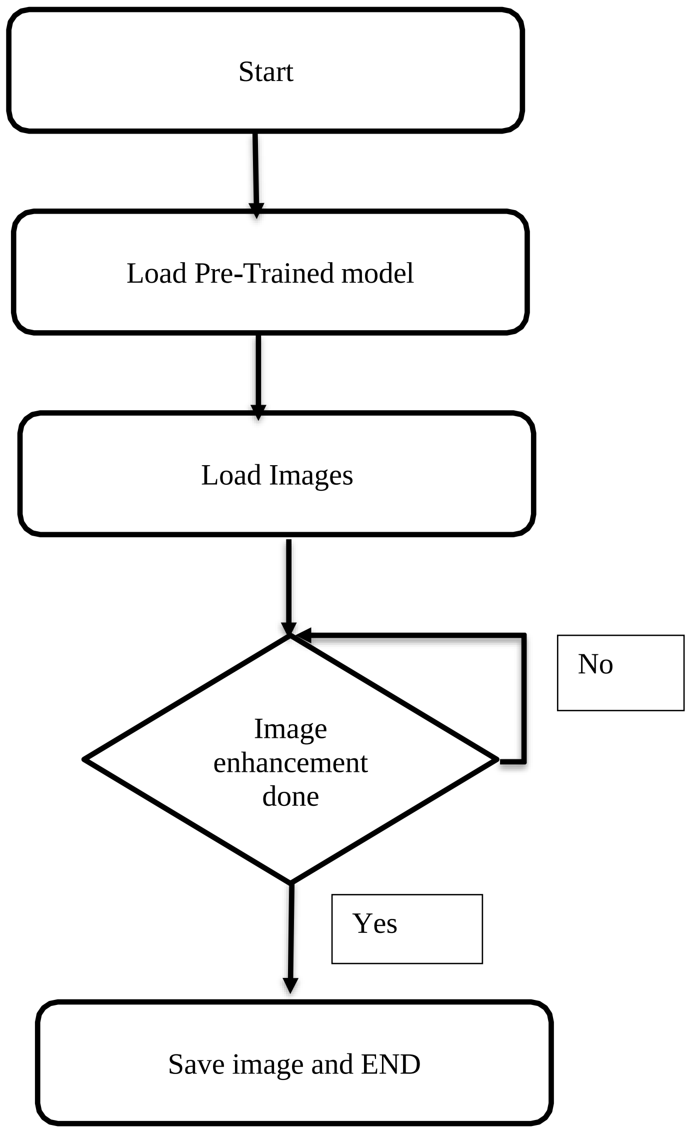

Fig 5.3: Flow diagram of User Interaction with GUI

The above flow diagram represents the actual execution of the project. This involves loading the pretrained model and the images from the input folder. Each image is selected and enhanced. It is then stored in the output folder. If all the images from the input folder are processed, the execution stops. Else the processing continues for the next image.

5.4 Modules

1. GUI (Graphical User Interface)
2. DATA LOADER
3. ABSGN MODEL
4. TRAIN AND TESTING
5. LOSS FUNCTION

5.4.1 GUI (Graphical User Interface)

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

Importing the module – tkinter

Create the main window (container)

Add any number of widgets to the main window

Apply the event Trigger on the widgets.

5.4.2 Data Loader

Data storage is often slow, in particular due to access latency, we want to parallelize data loading. But as the many things Python is well loved for do not include easy, efficient, parallel processing, we will need multiple processes to load our data, in order to assemble them into batches: tensors that encompass several samples. This is rather elaborate; but as it is also relatively generic, PyTorch readily provides all that magic in the Data Loader class. Its instances can spawn child processes to load data from a dataset in the background so that it's ready and waiting for the training loop as soon as the loop can use it. We will meet and use Dataset and Data Loader

In our training code, we chose minibatches of size 1 by picking one item at a time from the dataset. The torch. utils.data module has a class that helps with shuffling and organizing the data in minibatches: Data Loader. The job of a data loader is to sample minibatches from a dataset, giving us the flexibility to choose from different sampling strategies. A very common strategy is uniform sampling after shuffling the data at each epoch. Figure 7.14 shows the data loader shuffling the indices it gets from the Dataset

5.4.3 Absgn Model

ABSGN uses wavelet transform to transform the main feature map to three smaller scales. At the lowest resolution layer, we offer a Global Spatial Attention (GSA) Block including spatial attention module to be aware of the global context/color information. Apart from the lowest solution level, we add Multi-level Guided Densely Block (MGDB) one or two blocks

5.4.4 Train And Testing

- Objective: The main objective of this module is to test and train the Neural network of the model
- Input: Image is given as input in the form of 32 * 32 matrix is transferred to model to process.
- Output: Enhancement of image in each matrix and training of model.
- PyTorch is a python framework and module which can be used to train and test a model at architectural or Neural layer of the network.
- The image is tested and trained to know the efficiency of the model and to check whether the proposed system is working fine or not.

5.4.5 Loss Function

- The main objective of this module is to test the accuracy of the model, Both Image input and output is given as input to compare both the images The accuracy of the model is known by factors like PSNR, SSIM, MAE. PyTorch is a python framework and module which can be used to train and test a model at architectural or Neural layer of the network.

CHAPTER – 06

IMPLEMENTATION AND SAMPLE CODE

6.1 Languages/Tools/API's used

6.1.1 Python

Programming language used to build the system is Python. Python is an interpreted, object oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. It is flexible, has a great library ecosystem, versatile, has low entry barriers, easy to understand, popular and has a huge community of users, hence, Python is the preferred choice.

6.1.2 PyTorch

Convolutional neural network is developed using PyTorch in the proposed system. PyTorch is an optimized tensor library primarily used for Deep Learning applications using GPUs and CPUs. It is an open-source machine learning library for Python, mainly developed by the Facebook AI Research team. It is one of the widely used Machine learning libraries, others being TensorFlow and Keras. PyTorch library is relatively highly efficient compared to TensorFlow and Keras, hence, PyTorch turns out to be a preferable choice.

6.1.3 Tkinter

Tkinter is used to build GUI for the proposed system. Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run. Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes Tkinter a compelling choice for building GUI applications in Python.

6.1.4 OpenCV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.

6.2 Validation methodology

Image enhancement or improving the visual quality of a digital image can be subjective. Saying that one method provides a better-quality image could vary from person to person. For this reason, it is necessary to establish quantitative/empirical measures to compare the effects of image enhancement algorithms on image quality. In the proposed system we have used the following metrics to evaluate the model:

6.2.1 MAE (Mean Absolute Error)

It is the Difference between original and enhanced image and is given as

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i|$$

As the name suggests, the mean absolute error is an average of the absolute errors ei = |fi-yi |, where fi is the prediction and yi is the true value. Note that alternative formulations may include relative frequencies as weight factors. The mean absolute error used the same scale as the data being measured. This is known as a scale-dependent accuracy measure and therefore cannot be used to make comparisons between series using different scales.

Or simply

$$\mathbf{MAE = | E(x) – E(y) 1|}$$

Where E(x)= average intensity of input image E(y)= average intensity of enhanced image

6.2.2 SSIM (Structural Similarity Index)

The Structural similarity (SSIM) metric aims to measure quality by capturing the similarity of images. Three aspects of similarity: Luminance, contrast and structure is determined and their product is measured. Luminance comparison function l (X, Y) for reference image X and test image Y is defined as below

$$L(X,Y) = \frac{2\mu x\mu y + c1}{\mu x + \mu y + c1}$$

Where µx and µy are the mean values of X and Y respectively and C1 is the stabilization constant. Similarly, the contrast comparison function c (X, Y) is defined as

$$c(X,Y) = \frac{2\sigma_x\sigma_y + C2}{\sigma_x^2 + \sigma_y^2 + C2}$$

Where the standard deviation of X and Y are represented as σx and σy and C2 is the stabilization constant. The structure comparison function s (X, Y) is defined as

$$\text{S (X, Y)} = \frac{\boldsymbol{\sigma xy^{+} c3}}{\boldsymbol{\sigma x\sigma y + c3}}$$

Where σxy represents correlation between X and Y and C3 is a constant that provides stability. By combining

$$\text{SSIM (X, Y)} = [\, l\,(X, Y)]^{\boldsymbol{\partial}} \,.\, [(c\,(X, Y)]^{\beta} .\, [(s\,(X, Y)]^{\gamma}$$

the three comparison functions, The SSIM index is obtained as below

and the parameters are set as a = β = ϒ = 1 and C3=C2/2 From the above parameters the SSIM index can be defined as

$$SSIM(X,Y) = \frac{(2\mu_X\mu_Y + C1)(2\sigma_{xy} + C2)}{(\mu_x^2 + \mu_y^2 + C1)(\sigma_x^2 + \sigma_y^2 + C2)}$$

Symmetric Gaussian weighting functions are used to estimate local SSIM statics. The mean SSIM index pools the spatial SSIM values to evaluate overall image quality.

$$SSIM(X,Y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j - y_j)$$

Where xj and yj are image patches covered by the jth window and the number of local windows over the image are represented by M.

6.2.3 PSNR (Peak Signal to Noise Ratio)

The term peak signal-to-noise ratio (PSNR) is an expression for the ratio between the maximum possible value of a signal and the power of distorting noise that affects the quality of its representation. Because many signals have a very wide dynamic range, i.e., ratio between the largest and smallest possible values of a changeable quantity, the PSNR is usually expressed in terms of the logarithmic decibel scale. The higher the PSNR, the better degraded image has been reconstructed to match the original image and the better the reconstructive algorithm. This would occur because we wish to minimize the MSE between images with respect to the maximum signal value of the image. The mathematical representation of the PSNR is as follows:

$$\mathbf{PSNR = 20 \log_{10} \left(\frac{MAXf}{\sqrt{MSE}}\right)}$$

f represents the matrix data of our original image

MAXf is the maximum signal value that exists in our original “known to be good” image

MSE is the mean squared error.

The higher the PSNR, the better degraded image has been reconstructed to match the original image and the better the reconstructive algorithm.

6.3 Sample Code

```
import numpy as np
import os,cv2,time,easygui,sys,glob,dataloader
import torch,torchvision,argparse,model
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import tkinter as tk
import tkinter.messagebox
import tkinter.font as font
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import torch.nn as nn
import torchvision
import torch.backends.cudnn as cudnn
import torch.optim
from torchvision import transforms
from PIL import Image

originalImage = None

def upload():
    global originalImage,tkImage
    imagePath = easygui.fileopenbox() #image path
    if imagePath==None:
        return
    originalImage = cv2.imread(imagePath) #open image
    originalImage = cv2.resize(originalImage, (600, 400)) #resize
image
    originalImage = cv2.cvtColor(originalImage,cv2.COLOR_BGR2RGB)
    tkImage  =  Image.fromarray(originalImage)  #convert  to  PIL
format
    tkImage  =  ImageTk.PhotoImage(tkImage)  #convert  to  Imagetk
format
    Label(right_frame,  image=tkImage).grid(row=0,column=0,  padx=
3, pady=3) #display image in right frame
    IMAGE=cv2.imread(imagePath)
    os.chdir(r"C:\Users\CHARISH  PATEL  M  N\Desktop\Afinal  year
projects\LowlightProject\data\test_data\images")
    newName="Inputimage.jpg"
    cv2.imwrite(newName, IMAGE)
```

```
def lowlight(image_path):
    global originalImage2,tkImage2
    os.environ['CUDA_VISIBLE_DEVICES']='0'
    data_lowlight = Image.open(image_path)
    data_lowlight = (np.asarray(data_lowlight)/255.0)
    data_lowlight = torch.from_numpy(data_lowlight).float()
    data_lowlight = data_lowlight.permute(2,0,1)
    data_lowlight = data_lowlight.unsqueeze(0)
    DCE_net = model.enhance_net_nopool()
    DCE_net.load_state_dict(torch.load(r'C:\Users\CHARISH PATEL M
N\Desktop\Afinal      year      projects\LowlightProject\snapshots
\Epoch99.pth', map_location={'cuda:0': 'cpu'}))
    start = time.time()
    _,enhanced_image,_ = DCE_net(data_lowlight)

    end_time = (time.time() - start)
    print(end_time)
    # image_path = image_path.replace('test_data','result')
    # result_path = image_path
    #                            if                              not
os.path.exists(image_path.replace('/'+image_path.split("/")
[-1],'')):
      #
os.makedirs(image_path.replace('/'+image_path.split("/")[-1],''))

    torchvision.utils.save_image(enhanced_image,        r"C:\Users
\CHARISH  PATEL  M  N\Desktop\Afinal  year  projects\LowlightProject
\data\result\images\output.jpg")
    imagePath2  =  (r"C:\Users\CHARISH  PATEL  M  N\Desktop\Afinal
year     projects\LowlightProject\data\result\images\output.jpg")
#image path
    originalImage2 = cv2.imread(imagePath2) #open image
    originalImage2   =   cv2.resize(originalImage2,   (600,   400))
#resize image
    originalImage2                                               =
cv2.cvtColor(originalImage2,cv2.COLOR_BGR2RGB)
    tkImage2  =  Image.fromarray(originalImage2)  #convert  to  PIL
format
    tkImage2  =  ImageTk.PhotoImage(tkImage2)  #convert  to  Imagetk
format
    Label(mid_frame, image=tkImage2).grid(row=0,column=0, padx=3,
def image_enhancement():
# test_images
    with torch.no_grad():
        #   filePath   =   (r'F:\6thsem\AIML\Lab\GUI\Newfolder\data
\test_data')
        # file_list = os.listdir(filePath)
        # for file_name in file_list:
        test_list   =   glob.glob(r"C:\Users\CHARISH   PATEL   M   N
\Desktop\Afinal    year    projects\LowlightProject\data\test_data
\images\*")
        for image in test_list:
            print(image)
            lowlight(image)

def savefile():
    imagePath3  =  (r"C:\Users\CHARISH  PATEL  M  N\Desktop\Afinal
year     projects\LowlightProject\data\result\images\output.jpg")
#image path
    originalImage3 = cv2.imread(imagePath3)
    originalImage3                                                  =
cv2.cvtColor(originalImage3,cv2.COLOR_BGR2RGB)
    edge = Image.fromarray(originalImage3)
    filename          =         filedialog.asksaveasfile(mode='w',
defaultextension=".jpg")
    if not filename:
        return
    edge.save(filename)

def Close():
    top.destroy()

top=tk.Tk()

window_width = 2200

window_height = 700

# get the screen dimension
screen_width = top.winfo_screenwidth()
screen_height = top.winfo_screenheight()
```

```
# find the center point
center_x = int(screen_width/2 - window_width / 2)
center_y = int(screen_height/2 - window_height / 2)

#                                    top.geometry(f'{window_width}
x{window_height}+{center_x}+{center_y}')

icon_img = ImageTk.PhotoImage(Image.open(r"C:\Users\CHARISH PATEL
M N\Desktop\Afinal year projects\LowlightProject\download.png"))
top.iconphoto(False, icon_img)
top.title('Image enhancement using deep learning')

top.config(bg = "grey16")


left_frame = Frame(top, width=300, height=530, bg='grey16')
left_frame.grid(row=1, column=0, padx=10, pady=55)
mid_frame  =  Frame(top,  width=700,  height=530,  bg='light  slate
grey')
mid_frame.grid(row=1, column=2, padx=15, pady=25)
right_frame  =  Frame(top,  width=700,  height=530,  bg='light  slate
grey')
right_frame.grid(row=1, column=1, padx=15, pady=25)
button_frame = Frame(top, width=18, height=5, bg='grey16')
button_frame.grid(row=2, column=1, padx=0, pady=0)


ai_image = ImageTk.PhotoImage(Image.open(r"C:\Users\CHARISH PATEL
M N\Desktop\Afinal year projects\LowlightProject\index.jpg"))
Label(right_frame,  image=ai_image).grid(row=0,column=0,  padx=3,
pady=3)

a_image  =  ImageTk.PhotoImage(Image.open(r"C:\Users\CHARISH  PATEL
M N\Desktop\Afinal year projects\LowlightProject\index.jpg"))
Label(mid_frame,    image=a_image).grid(row=0,column=0,    padx=3,
pady=3)

button_font = font.Font(family='Times', size=11)

upload_button      =      tk.Button(left_frame,      text="UPLOAD
ai_image = ImageTk.PhotoImage(Image.open(r"C:\Users\CHARISH PATEL
M N\Desktop\Afinal year projects\LowlightProject\index.jpg"))
Label(right_frame,  image=ai_image).grid(row=0,column=0,  padx=3,
pady=3)

a_image  = ImageTk.PhotoImage(Image.open(r"C:\Users\CHARISH  PATEL
M N\Desktop\Afinal year projects\LowlightProject\index.jpg"))
Label(mid_frame,    image=a_image).grid(row=0,column=0,    padx=3,
pady=3)

button_font = font.Font(family='Times', size=11)

upload_button      =      tk.Button(left_frame,      text="UPLOAD
IMAGE",command=upload, fg="black",height=2,width=20,border='0')
upload_button.grid(row=0,column=1, padx=10, pady=15)
upload_button['font'] = button_font

enhance_button      =      tk.Button(left_frame,     text="ENHANCE
IMAGE",command=image_enhancement,fg="black",height=2,width=20)
enhance_button.grid(row=1,column=1, padx=0, pady=15)
enhance_button['font'] = button_font

save_button        =        tk.Button(left_frame,        text="SAVE
IMAGE",command=savefile,fg="black",height=2,width=20)
save_button.grid(row=4,column=1, padx=0, pady=15)
save_button['font'] = button_font

exit_button                  =                  tk.Button(left_frame,
text="EXIT",command=Close,fg="black",height=2,width=20)
exit_button.grid(row=5,column=1, padx=0, pady=15)
exit_button['font'] = button_font

top.mainloop()
```

CHAPTER – 07

TESTING

We first introduce the training details and provide experimental results on different datasets. Then we compare ABSGN with several state-of-the-art low-light image enhancement approaches.

7.1 Experimental Setting

Low-Light (LOL) dataset is a publicly available dark-light paired images dataset in the real sense. The lowlight images are collected by changing exposure time and ISO. It contains 500 images in total, The resolution of each of these images is 400 × 600. we use 485 images of them for training, and the rest for evaluation as suggested by Most state-of-the-art low-light image enhancement solutions select the LOL as their training dataset. To train our ABSGN model, we also use LOL dataset as training and validation dataset for low-light image enhancement task. The LOL dataset consists of two parts: the real-world dataset and the synthetic dataset. Because the LOL synthetic dataset cannot simulate the degradation of real-world images very well, we deprecate the synthetic dataset where low-light images are synthesized from normal-light images and only use the real-world dataset.

7.2 Dataset Comparison

Our datasets for comparison include NPE, LIME MEF, DICM which are used by some recent low-light image enhancement networks. For these datasets lack of reference images, we use non-reference metric including No-reference Image Quality and Uncertainly Evaluator (UNIQUE) and Blind/Reference less Image Spatial Quality Evaluator (BRISQUE). In addition, we also choose another commonly used data set called MIT-Adobe FiveK dataset for training and evaluation. MIT-Adobe FiveK contains 5000 images of various indoor and outdoor scenes captured with DSLR cameras in different lighting conditions. The tonal attributes of all images are manually adjusted by five different trained photographers (labelled as experts A to E). we choose the enhanced images of expert C as the ground truth. Moreover, the first 4500 images are used for training and the last 500 for testing. When training our model, we randomly crop 256 × 256 patches from the training images.

The input patches of the proposed network are randomly flipped and rotated for data augmentation. The parameters of network are Kaiming initialized. We train the whole network for 300 epochs overall.

7.3 Evaluation on LOL Dataset

We compare our proposed network with several state-of-the-art low-light image enhancement solutions: MSRCR, LIME, NPE, JED, MBLLEN, RetinexNet, GLADNet, RDGAN, KinD++, Zero-DCE, EnlightenGAN and MIRNet. To compare the performance, we determined to take above measures as the objective and subjective measurements. Our method achieves the best performance in PSNR, SSIM, LPIPS and FSIM, MIRNet is the second-best method, Although MIRNet adopts more complicated structure and has more than ten times the number of parameters, ABSGN is still able to achieve better PSNR and SSIM than MIRNet on average.

We can see that from Table 7.1 our ABSGN shows the best PSNR and SSIM in different light levels. There also show some detail results of several lowlight image enhancement, where KinD++ is a classical decomposing network and MIRNet has excellent feature extraction capabilities. We can see all the three low-light image enhancement networks are able to achieve an obvious improvement compared with low light images.

7.4 Comparison on NPE, LIME, MEF, DICM

Table shows the comparison on NPE, LIME, MEF and DICM datasets. Our model has the best average UNIQUE and the second average BRISQUE. From figure, we can also conclude a similar conclusion to the NPE, LIME, MEF, DICM datasets. At a darker region, restoring networks tend to smooth the noise too much, because the network is difficult to distinguish true details from the dark region. ABSGN can better preserve details at dark regions, such as soils and floors.

Method	**NPE**	**LIME**	**MEF**	**DICM**	**Average**
Dark	0.793/19.81	0.826/21.81	0.738/23.56	0.795/21.57	0.788/21.68
PIE	0.801/21.72	0.791/22.72	0.752/11.02	0.791/21.72	0.783/19.30
LIME	0.786/18.24	0.774/20.44	0.722/15.25	0.758/23.48	0.760/19.35
MBLLEN	0.793/34.46	0.768/30.26	0.717/37.44	0.787/32.44	0.766/33.65
RetinexNet	0.828/16.04	0.794/31.47	0.755/20.08	0.770/29.53	0.786/24.48
KinD	0.792/19.65	0.766/39.29	0.747/31.36	0.776/32.71	0.770/30.75
Zero-DCE	0.814/17.06	0.811/21.40	0.762/16.84	0.777/27.35	0.791/20.66
MIRNet	0.815/29.35	0.814/33.73	0.768/21.45	0.812/33.71	0.804/29.65
OURS	**0.823/29.73**	**0.827/32.23**	**0.784/38.52**	**0.801/33.23**	**0.806/30.83**

Table 7.1: Loss Function Comparison with Existing System

7.5 ABGSN Testing

We train the whole network for 300 epochs overall. The learning rate is initialized as 1 × 10−4 at the first 200 epochs and reduce to 5 × 10−5 in the next 50 epochs. We finetune our model at the last 50 epochs with a 1 × 10−5 learning rate. For optimization, we use Adam optimizer with β1 = 0.5, β2 = 0.999 and batch size equals to 5. We use L1 loss and SSIM loss for the total loss function, L1 loss is a PSNR-oriented optimization in the training process. SSIM loss can keep the overall structure. We adopt PSNR, SSIM, LPIPS and FSIM as the quotative metrics to measure the performance of our method.

Method	SSIM	PSNR	LPIPS	FSIM
Input	0.194	7.77	0.4173	0.7190
MSRCR	0.4615	13.17	0.4404	0.8450
LIME	0.4449	16.76	0.4183	0.8549
NPE	0.4839	16.97	0.4156	0.8964
JED	0.6509	13.69	0.3578	0.8812
MBLLEN	0.7247	17.86	0.3672	0.9262
RetinexNet	0.4246	16.77	0.4670	0.8641
GLADNet	0.6820	19.72	0.3994	0.9329
RDGAN	0.6357	15.94	0.3985	0.9276
KinD++	0.8203	21.30	0.1645	0.9456
Zero-DCE	0.5623	14.87	0.3852	0.9276
EnlightenGAN	0.6515	17.48	0.3903	0.9226
MIRNet	0.8321	24.14	0.0846	0.9547
Ours	0.8680	24.59	0.0772	0.9659

Table 7.2: Performance Comparison with Existing systems

7.6 MGDB Layer Testing

We use the DenseRes Block to replace our MGDB to test the effect. In Addition, we try to apply the custom convolution to take place of all the dilation convolutions, which will tell the necessity of introducing dilation convolution. The results suggest that MGDB can get more local information and better performance in SSIM/PSNR. Similarity, we respectively use SPA module and GIA module substitute for our GSA module to prove the advantage of our proposed module. As is shown in Table, Compared to SPA module and GIA module, our module has irreplaceable advantage. These above modules can’t replace our GSA module to extract the global information.

Method	SPA	GIA	DenseRes	WDC	Ours
PSNR(Db)	22.30	23.42	22.67	23.16	24.59
SSIM	0.8456	0.8513	0.8483	0.8546	0.8675

Table 7.3: MGDB Layer Analysis

7.7 Testing Conclusion

We compare BSWN with other state-of-art deep learning low-light image enhancement methods in terms of the number of parameters, the inference time and UQI in LOL Dataset. As is shown in table, our model has the best UQI, meaning our results has closest information with the reference images.

Although the parameters of our model is a little heavy, MIRNet is ten times bigger than ours. What's more, inference speed of our model is very fast. As is shown in Figure, we tested all of the deep learning models on the same platform. The closer the method is to the upper left corner in the figure, the faster the model is and the higher the SSIM is. As is shown in the last column of table in table 5.

Deep Learning Method	**Params**	**Time Cost**	**UQI**
MBLLEN	1.95M	80ms	0.8261
RetinexNet	9.2M	20ms	0.9110
GLAD	11M	25ms	0.9204
RDGAN	4.2M	30ms	0.8296
Zero-DCE	0.97M	2ms	0.7205
EnlightenGAN	33M	20ms	0.8499
KinD++	35.7M	280ms	0.9482
MIRNet	365M	340ms	0.9556
Our Model	33M	14ms	0.9589

Table 7.4: Time Consumption Comparison with Existing System

Our model only takes 14 ms to process a real-world low-light image with resolution of 600x400 from the LOL dataset. The speed of our model ranks second among all of the compared methods. Although Zero-DCE has very fast inference speed, which is only to deal with a little darker image. the biggest problem of Zero-DCE is difficult to restore the dark image especially in the night, which leads to its lack of strong applicability and application prospects. our model applies self-guidance strategy to parallel deal with the feature map in different resolutions. This is why our model can run at such a high-speed keeping excellent restoring capacity.

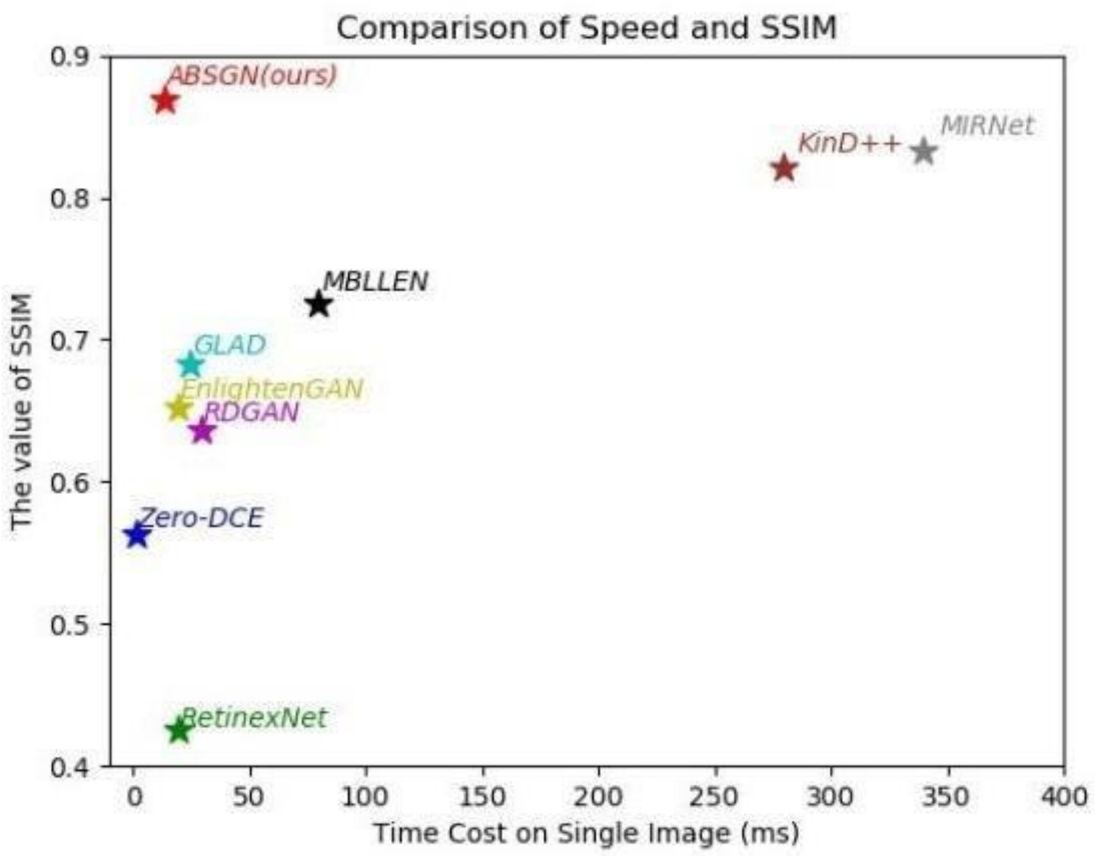

Fig 7.1: Comparison of speed and SSIM with Existing System

7.8 Result and Analysis

In this paper, we proposed an Attention based Broadly Self-guided Network (ABSGN) for low-light image enhancement. ABSGN adopts a top-down manner to restore the low-light images. We use wavelet transform and a Conv+PReLU layer to generate input variations with different spatial resolutions. At the lowest solution, we design GSA module to fully collect the global information. Then, we embed a MGDB into the middle two low spatial resolution levels to fully get local information. respectively, At the full resolution level, we employ more MGDB and a Dense connection to further reuse the information of the main feature map from the input image. The proposed ABSGN was validated on image restoration and real world low-light image enhancement benchmark and ABSGN is able to generate higher quality results than the compared state-of-the-art methods. Further, our ABSGN has excellent inference speed, which has good practical value and application prospects.

7.8.1 Results

For evaluation of the model's performance the second part of the SICE data set is used. Second part of the SICE dataset consists of 229 image sequences with 7 images in each sequence. In each sequence a second image is chosen, hence the total size of testing dataset size is 229 and each image is resized to 480x640. The model takes approximately 40s to enhance one image. Some output images are shown below in Fig 7.2

For the proposed system, the values obtained for peak signal to noise ratio (PSNR) is 17.56, structural similarity index (SSIM) is 0.63 and mean absolute error (MAE) is 94.7. The values of peak signal to noise ratio, structural similarity index and mean absolute error obtained for SRIE (Simultaneous Reflectance and Illumination Estimation) method.

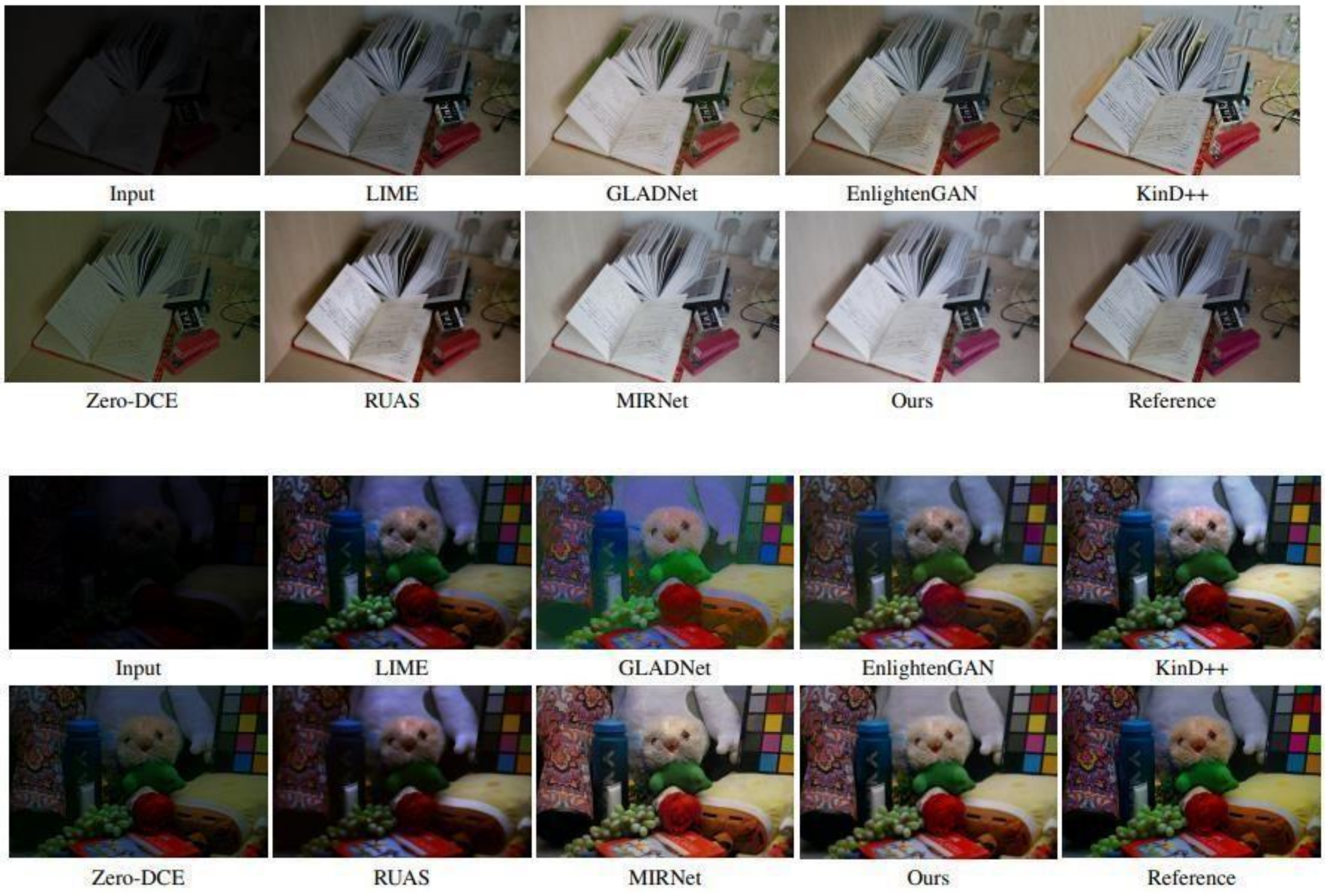

Fig 7.2: Result Analysis with Existing system image output

7.8.2 Analysis

We compare our proposed network with several state-of-the-art low-light image enhancement solutions: MSRCR [12], LIME, NPE, JED [29], MBLLEN [21], RetinexNet, GLADNet, RDGAN, KinD++, Zero-DCE, EnlightenGAN and MIRNet. To compare the performance, we determined to take above measures as the objective and subjective measurements. As shown in table 1, our method achieves the best performance in PSNR, SSIM, LPIPS and FSIM, MIRNet is the second-best method, Although MIRNet adopts more complicated structure and has more than ten times the number of parameters, ABSGN is still able to achieve better PSNR and SSIM than MIRNet on average.

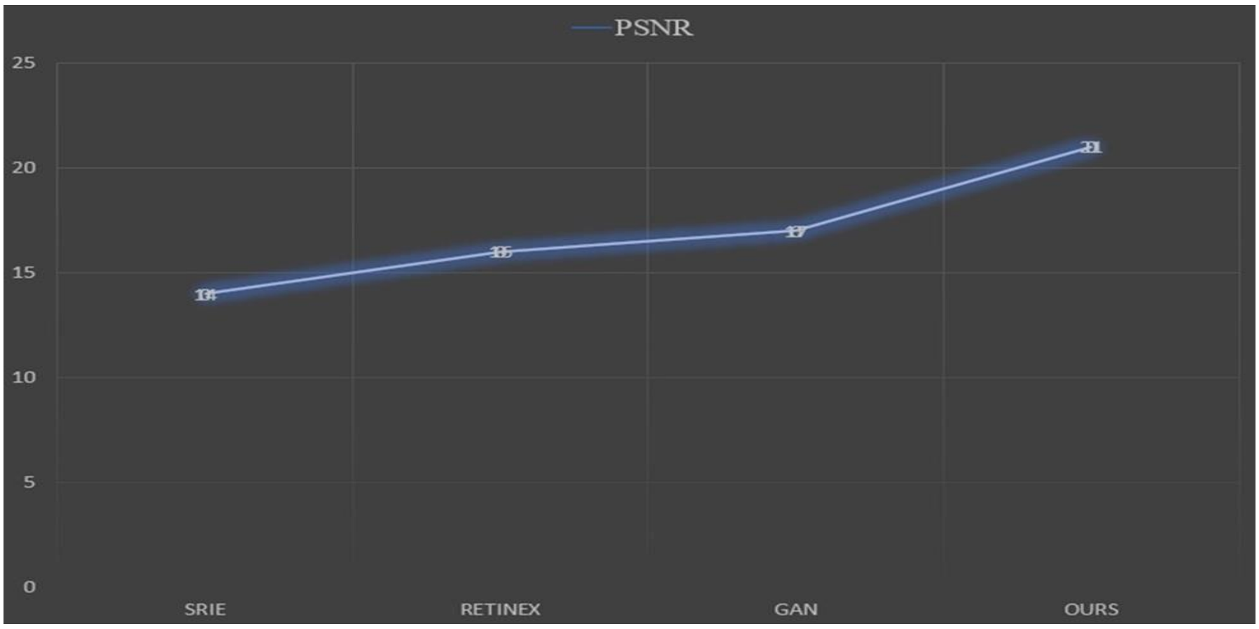

Fig 7.3: PSNR Result Analysis with existing Systems

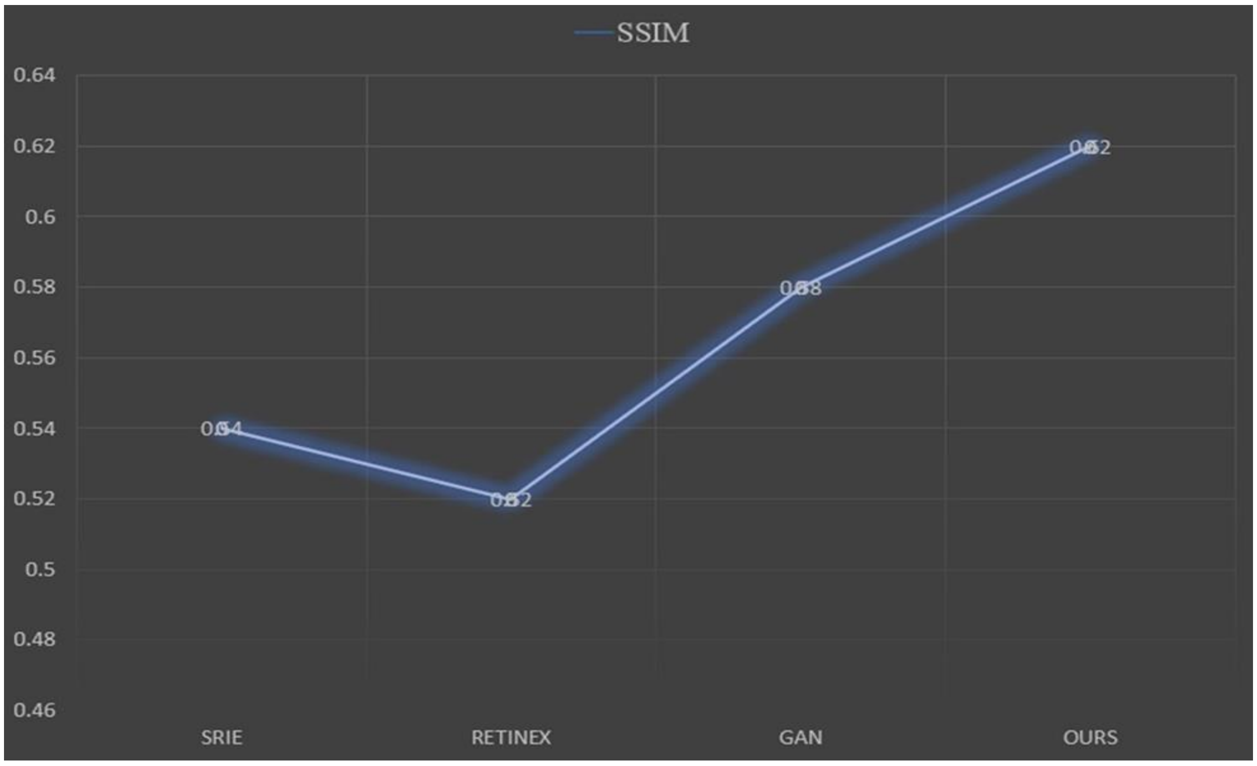

Fig 7.4: SSIM Result analysis with existing Systems

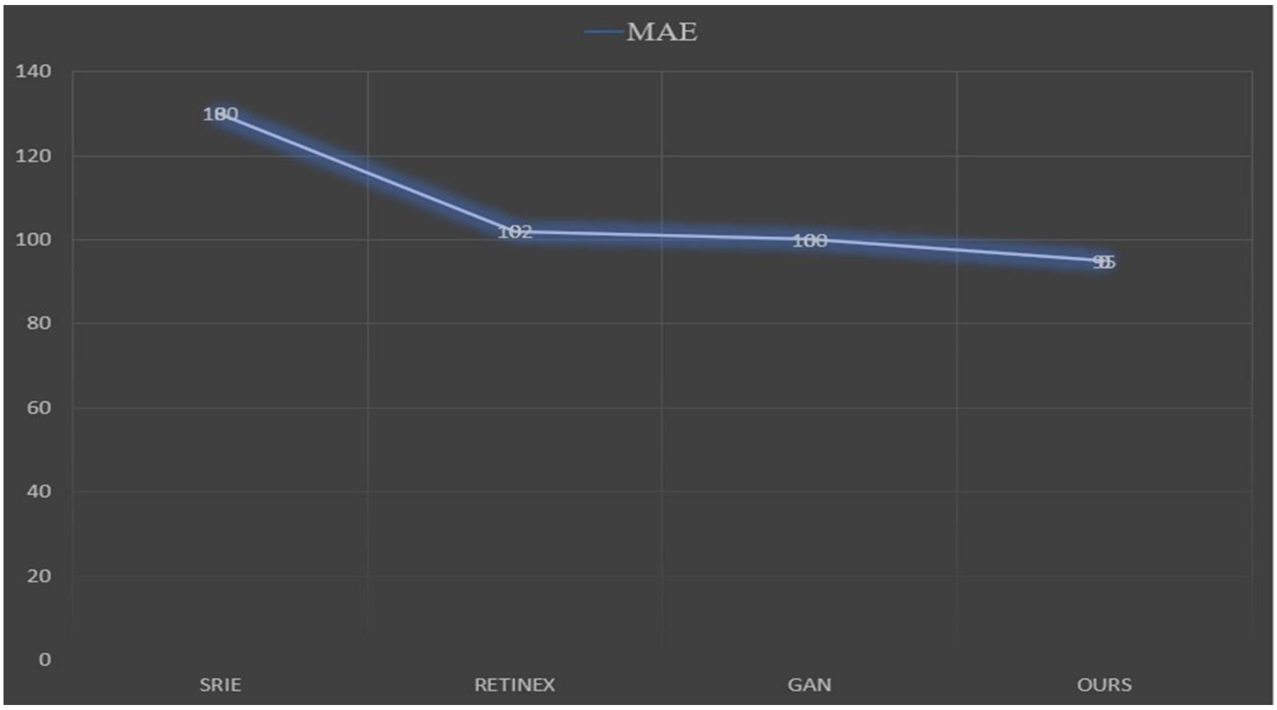

Fig 7.5: MAE Result Analysis with existing Systems

CHAPTER - 08

SNAPSHOTS

Below are some Snapshots of the Graphical user interface developed to enhance the lowlight image

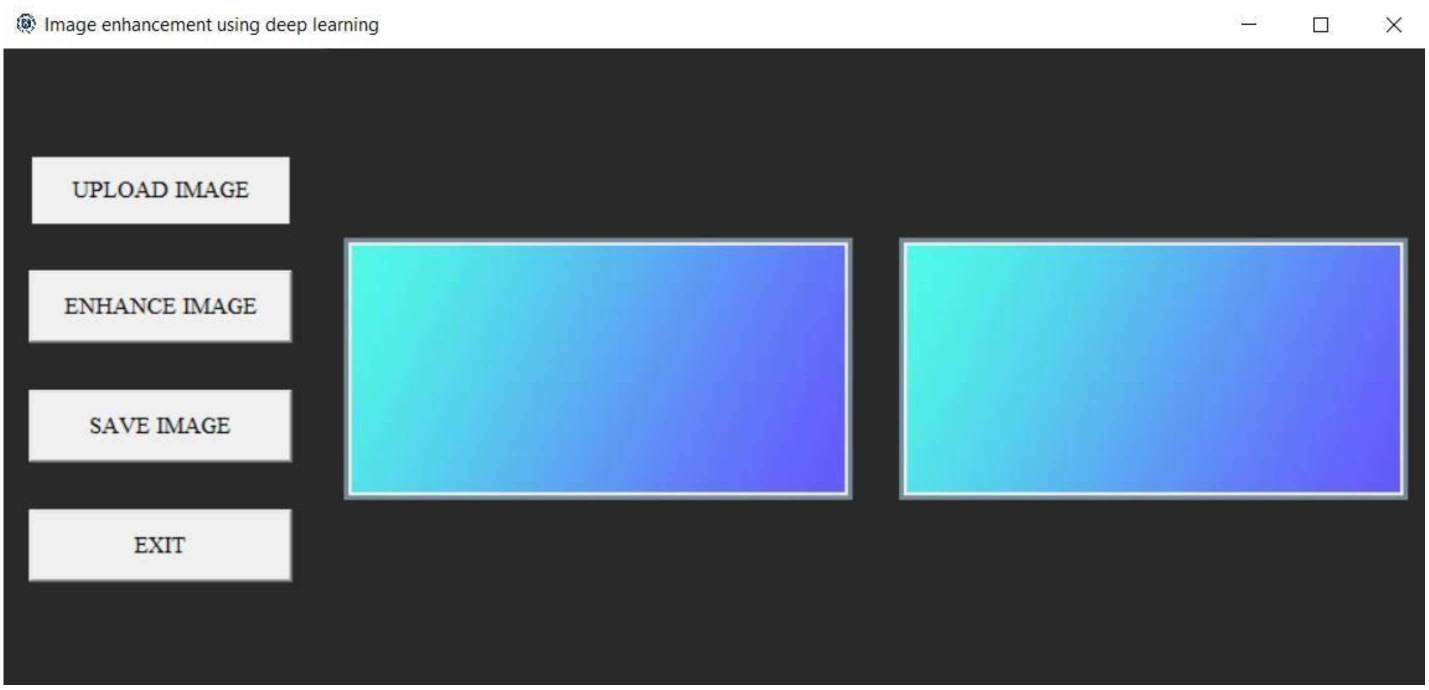

Fig 8.1: Graphical User interface Snapshot

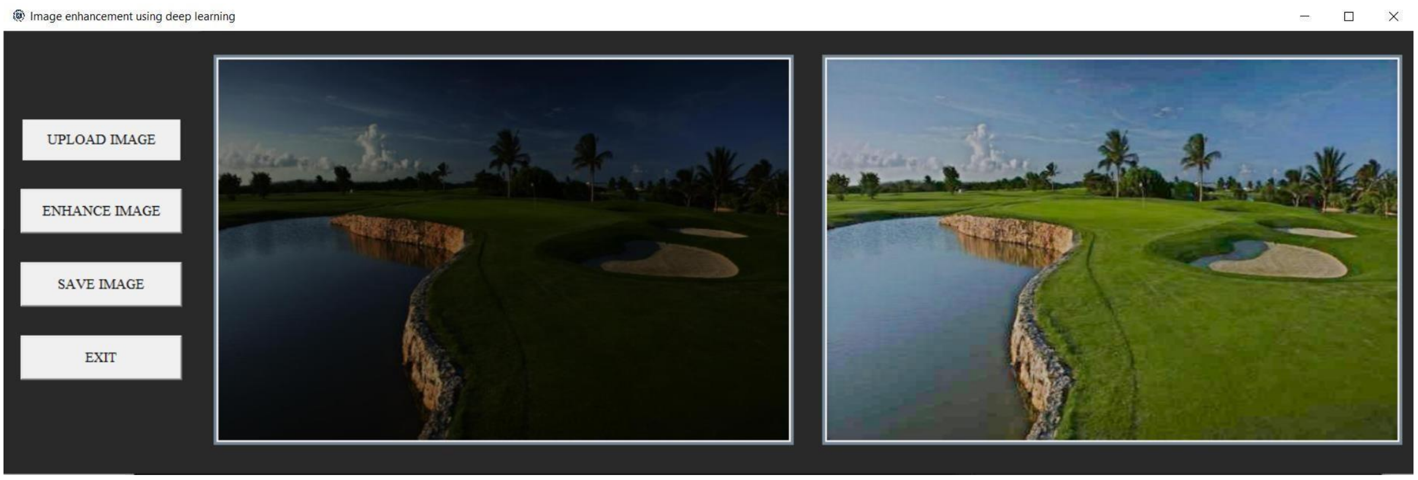

Fig 8.2: Graphical User interface Snapshot with output1

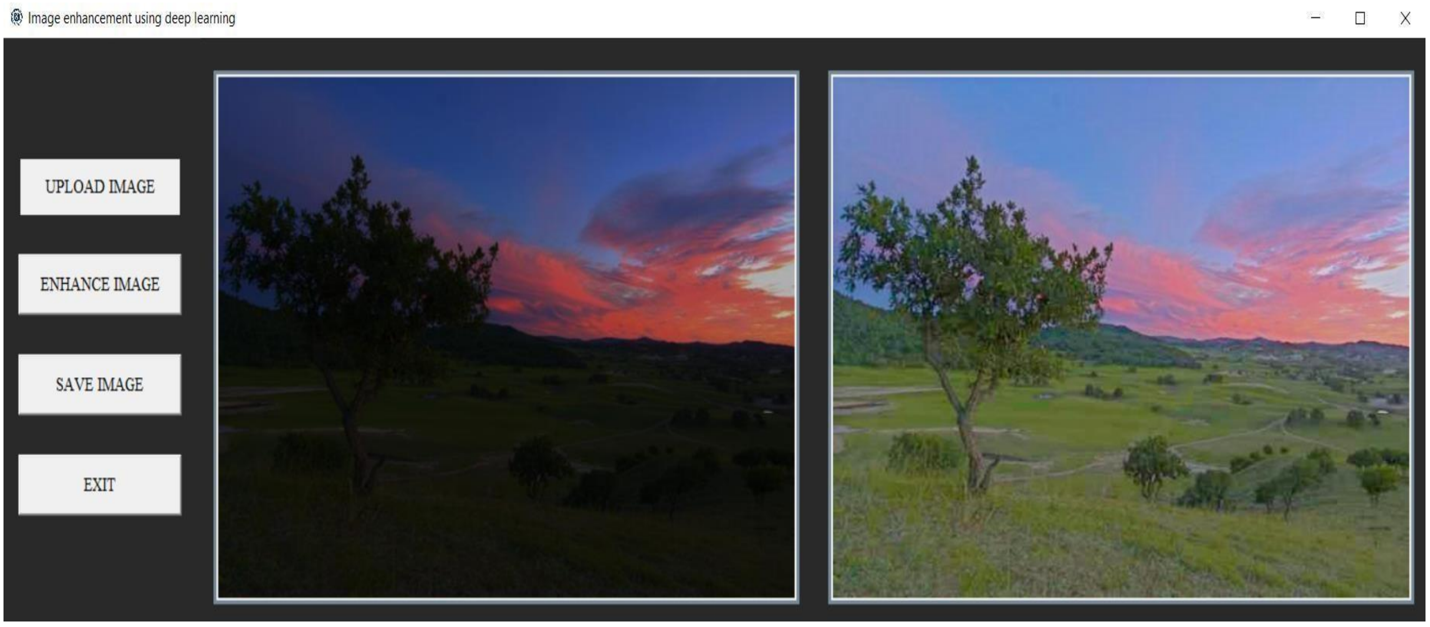

Fig 8.3: Graphical User interface Snapshot with output 2

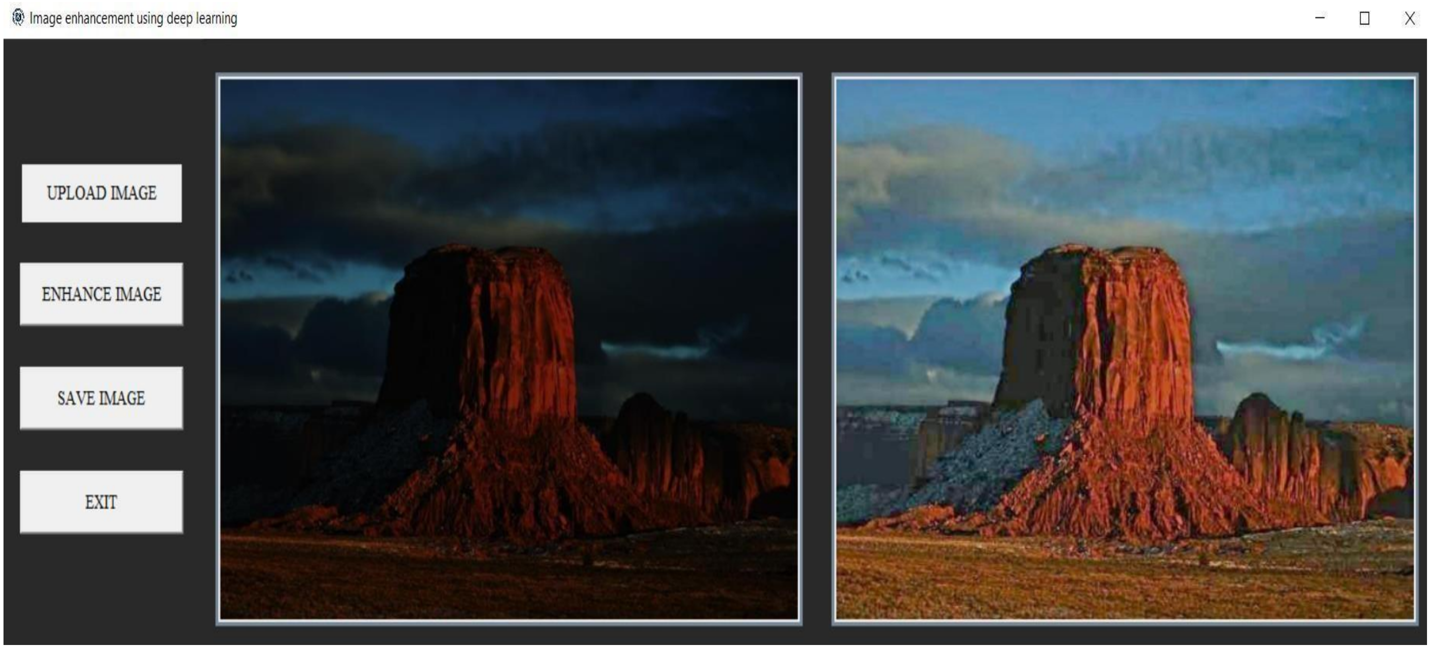

Fig 8.4: Graphical User interface Snapshot with output 3

CHAPTER – 09

FUTURE ENHANCEMENT

9.1 Novelty in the proposed solution

The proposed low-light enhancement network is independent of paired and unpaired training data, thus avoiding the risk of overfitting. As a result, the proposed method generalizes well to various lighting conditions. The proposed system works on the basis of an image-specific curve that is able to approximate pixel-wise and higher-order curves by iteratively applying itself. Such image-specific curves can effectively perform mapping within a wide dynamic range. Overall, the proposed method supersedes state-of-the-art performance both in qualitative and quantitative metrics. More importantly, it is capable of improving high-level visual tasks, e.g., face detection, without inflicting high computational burden.

9.2 Limitations of the project

Although the image enhancement is efficient enough as per requirements, there are a few cases of usage which could be added for extra features. Some of these limitations of the project are:

● The blurred images are not enhanced in this project.

● Enhancing is done entirely by machine, the user has no option to control any parameters with respect to image enhancement.

9.3 Future enhancements

The proposed project serves as a standalone convolutional neural network model for low light image enhancement. Many real time applications like CCTV surveillance systems, animal detection systems, theft detection systems etc., face the drawback of underperformance when used in low light conditions. The proposed model can be integrated with these Realtime systems as a pre-processing unit, which leads to the improvement in performance and proper utilization of the potential of the model

CHAPTER-10

CONCLUSION

ABSGN adopts a top-down manner to restore the low-light images. We use wavelet transform and a Conv+PReLU layer to generate input variations with different spatial resolutions. At the lowest solution, we design GSA module to fully collect the global information. Then, we embed a MGDB into the middle two low spatial resolution levels to fully get local information. respectively, At the full resolution level, we employ more MGDB and a Dense connection to further reuse the information of the main feature map from the input image. The proposed ABSGN was validated on image restoration and real world low-light image enhancement benchmark and ABSGN is able to generate higher quality results than the compared state-of-the-art methods. Further, our ABSGN has excellent inference speed, which has good practical value and application prospects. The proposed low-light enhancement network is independent of paired and unpaired training data, thus avoiding the risk of overfitting. As a result, the proposed method generalizes well to various lighting conditions. The proposed system works on the basis of an image-specific curve that is able to approximate pixel-wise and higher-order curves by iteratively applying itself. Such image-specific curves can effectively perform mapping within a wide dynamic range. This shows the potential of training a deep image enhancement model in the absence of reference images through task specific non-reference loss functions that indirectly evaluate enhancement quality. Overall, the proposed method supersedes state-of-the-art performance both in qualitative and quantitative metrics

CHAPTER - 11

BIBLIOGRAPHY

[1] Chunle Guo1, Chongyi Li1, Jichang Guo1, Chen Change Loy, Junhui Hou, Sam Kwong, Runmin Cong,” Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement”, Beijing Jiaotong University, 22 March 2020.

[2] Gershon Buchsbaum. A spatial processor model for object colour perception. J. Franklin Institute, 2019.

[3] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Fredo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In CVPR, 2021.

[4] Jianrui Cai, Shuhang Gu, and Lei Zhang. Learning a deep single image contrast enhancer from a multi-exposure image. IEEE Transactions on Image Processing, 2018.

[5] Chen Chen, Qifeng Chen, Jia Xu, and Koltun Vladlen. Learning to see in the dark. In CVPR, 2018.

[6] Yusheng Chen, Yuching Wang, Manhsin Kao, and Yungyu Chuang. Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In CVPR, 2018.

[7] Dinu Coltuc, Philippe Bolon, and Jean-Marc Chassery. Exact histogram specification. IEEE Transactions on Image Processing, 2021.

[8] Xueyang Fu, Delu Zeng, Yue Huang, Xiao-Ping Zhang, and Xinghao Ding. A weighted variational model for simultaneous reflectance and illumination estimation. In CVPR, 2016.

[9] Xiaojie Guo, Yu Li, and Haibin Ling. Lime: Low-light image enhancement via illumination map estimation. IEEE Transactions on Image Processing, 2017.

[10] Haidi Ibrahim and Nicholas Sia Pik Kong. Brightness preserving dynamic histogram equalization for image contrast enhancement. IEEE Transactions on Consumer Electronics, 2007.

[11] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In CVPR, 2018.

[12] Yifan Jiang, Xinyu Gong, Ding Liu, Yu Cheng, Chen Fang, Xiaohui Shen, Jianchao Yang, Pan Zhou, and Zhangyang Wang. EnlightenGAN: Deep light enhancement without paired supervision. In CVPR, 2019.

[13] Edwin H Land. The retinex theory of color vision. Scientific American, 2017.

[14] Chulwoo Lee, Chul Lee, and Chang-Su Kim. Contrast enhancement based on layered difference representation. In ICIP, 2018.

[15] Chulwoo Lee, Chul Lee, and Chang-Su Kim. Contrast enhancement based on layered difference representation of 2d histograms. IEEE Transactions on Image Processing, 2019.

[16] Chongyi Li, Chunle Guo, and Jichang Guo. Underwater image color correction based on weakly supervised color transfer. IEEE Signal Processing Letters, 2018.

Dataset: https://www.kaggle.com/