RV College of Engineering®

Autonomous institution affiliated to Visvesvaraya Technological University, Belagavi | Approved by AICTE, New Delhi

Go, change the world

End-to-End Deblurring with Denoising and Saturation Handling

A Minor Project Report (18EC64)

Submitted by,

Aritra Raychaudhuri	**1RV19EC028**
Moneet Mohan Devadig	**1RV19EC098**
Pragya Sen	**1RV19EC126**

Under the guidance of

Prof. Mahendra B.M.
Assistant Professor
Dept. of ECE
RV College of Engineering

In partial fulfillment of the requirements for the degree of

Bachelor of Engineering in

Electronics and Communication Engineering

2021-22

RV College of Engineering®, Bengaluru

(Autonomous institution affiliated to VTU, Belagavi)

Department of Electronics and Communication Engineering

.

CERTIFICATE

Certified that the minor project (18EC64) work titled ***End-to-End Deblurring with Denoising and Saturation Handling*** is carried out by **Aritra Raychaudhuri** (**1RV19EC028**), **Moneet Mohan Devadig** (**1RV19EC098**) and **Pragya Sen** (**1RV19EC126**) who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfillment of the requirements for the degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2021-22. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the minor project report deposited in the departmental library. The minor project report has been approved as it satisfies the academic requirements in respect of minor project work prescribed by the institution for the said degree.

Signature of Guide	Signature of Head of the Department	Signature of Principal
Prof. Mahendra B.M.	Dr. Ravish Aradhya H.V.	Dr. K. N. Subramanya

External Viva

Name of Examiners	Signature with Date
1.	
2.	

DECLARATION

We, **Aritra Raychaudhuri** , **Moneet Mohan Devadig** and **Pragya Sen** students of sixth semester B.E., Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, hereby declare that the minor project titled '**End-to-End Deblurring with Denoising and Saturation Handling**' has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** during the year 2021-22.

Further we declare that the content of the dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

Name Signature

1. Aritra Raychaudhuri(1RV19EC028)
2. Moneet Mohan Devadig(1RV19EC098)
3. Pragya Sen(1RV19EC126)

ACKNOWLEDGEMENT

We are indebted to our guide, **Prof. Mahendra B.M.**, Assistant Professor, RV College of Engineering . for the wholehearted support, suggestions and invaluable advice throughout our project work and also helped in the preparation of this thesis.

We also express our gratitude to our panel members **Prof. Usha Rani**, Associate Professor and **Prof. Sahana B**, Assistant Professor, Department of Electronics and Communication Engineering for their valuable comments and suggestions during the phase evaluations.

Our sincere thanks to the project coordinators **Ms. Sindhu Rajendran**, **Dr Nithin M** and **Dr Veena Devi** for their timely instructions and support in coordinating the project.

Our gratitude to **Prof. Narashimaraja P** for the organized latex template which made report writing easy and interesting.

Our sincere thanks to **Dr. H V Ravish Aradhya**, Professor and Head, Department of Electronics and Communication Engineering, RVCE for the support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** and our beloved Vice Principal, **Dr. K. S. Geetha** for the appreciation towards this project work.

We thank all the teaching staff and technical staff of Electronics and Communication Engineering department, RVCE for their help.

Lastly, we take this opportunity to thank our family members and friends who provided all the backup support throughout the project work.

ABSTRACT

This study deals with the deblurring and denoising of blurred images added with noise and outliers such as saturation.

Image blur is a very common occurence in natural photos, arising from different factors such as object motion, camera lens, out-of-focus, and camera shake. Deblurring is the process of removing blurs and restoring the high-quality latent image. Various types of blur exist including Motion blur, Gaussian blur, Average blur, Defocus blur etc. Images can have natural or synthetic blur.

Saturation describes the intensity of the color in an image. A grayscale or black-and-white photo has no color saturation, while a full-color photo of a field of sunlit wildflowers might be extremely saturated. In this study, saturation in images is also dealth with.

Image noise is random variation of brightness or color information in images. Denoising is the process of removing the noise present in images. It is seen that saturation and noise deeply affects the performance of deblurring networks. Motion blur and signal noise are the most important factors which causes image degradation. It is seen that in low light, image quality tends to be a tradeoff between motion blur and signal noise.

Existing deblurring techniques rely on implicit or explicit sharp edge restoration for blur kernel estimation. Though recently numerous deep learning methods have been developed with low computation costs, namely Generative Adversarial Networks and Convolutional Neural Networks, a model for handling deblurring along with the problem of saturation with a good level of accuracy does not currently exist.

The goal is to realise a new and improved GAN model for end-to-end deblurring in combination with saturation handling and denoising networks that can achieve more favourable values of image quality metrics(such as SSIM and PSNR) in comparison with existing models.

Comparing the performance of GAN and CNN, it is seen that GAN performs much better when it comes to image deblurring. For the proposed model, Super Resolution GAN (SRGAN) is chosen. It is seen that CNN tends to saturate accuracy as the depth of the model increases, which is not seen in SRGAN.

Existing Saturation handling methods make use of Confidence Maps and Encoder-Decoder Structures. GAN has been seen to provide better results than the Encoder-

Decoder structure when it comes to upsampling and it connects better to the SRGAN structure used in the deblurring process. In this work, Enahnced Super Resolution-GAN (ESRGAN) has been used for saturation handling as it has been seen to be more effective and can produce more realistic texture details than other super-resolution methods.

The proposed model aims to use Denoising Auto-Encoders to build a realistic noise model and combine it with an optimal deblurring and saturation handling network.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction to Deblurring and Denoising of images with outliers

CHAPTER 1

INTRODUCTION TO DEBLURRING AND DENOISING OF IMAGES WITH OUTLIERS

1.1 Introduction

Though recently numerous deep learning methods have been developed with low computation costs, a model for handling denoising and deblurring along with the problem of saturation with a good level of accuracy does not currently exist.

This study deals with realizing an optimal GAN model that can handle outliers as well as denoising, which is better than state of the art models. Previously CNN (Convolutional Neural Network) approaches have been extensively studied for the purpose of deblurring. While CNN is a fast and high-performance option, it has drawbacks like as the depth of the model increases, the accuracy starts to decrease(known as the degradation problem) and struggles when there is tilt or rotation in the blurred image. To avoid these issues, GAN (Generative Adversarial Networks) was chosen for deblurring. In the proposed model, Super Resolution GAN (SRGAN) [1] is used.

Confidence maps, Robust GANs, etc. are existing methods used for handling outliers. In this project, a modified version of ESRGAN (Enhanced Super Resolution GAN) [2] has been used for the same purpose due to better performance.

For denoising, Autoencoders[3] were chosen as they perform better than other methods when it comes to denoising.

1.2 Motivation

It has been observed that in the existing work in the area of image deblurring, not very satisfactory results for deblurring images with noise and outlier such as saturation have been obtained. For receiving presentable results, many constraints were placed on the amount of blur or noise the input could have and would result in ripples or ringing artifacts in the output image if the amount of noise and blur were not within the threshold and the current threshold value is not good enough for most real-life scenarios.

Deblurring of images plays an extremely important role in many aspects of life. For example, deblurring of images and videos from CCTV footage can be life-changing for a victim in pursuit of justice or deblurring of old photos and videos to be able to preserve

them for longer are few of the reasons this topic was chosen. In most of these cases, the noise or the blur in the input would not be within the threshold that existing models allow due to which it was decided to incorporate deblurring, denoising and saturation networks into one to realise an optimal model and produce better results than currently existing approaches.

1.3 Problem statement

Deblurring and denoising images with outliers using the help of an optimum GAN module.

1.4 Objectives

The objectives of the project are

1. To use the optimum GAN module for deblurring images.
2. To handle saturation while deblurring.
3. To use traditional denoising networks for image denoising.
4. To combine all the three architectures to obtain the final working model.

1.5 Literature Review

Generative Adversarial Networks (GANs) are popular as they have the ability to reproduce images. Many different GAN approaches were explored to understand the working, advantages, and disadvantages of the different GAN architectures.

Orest Kupyn et al. [4] present DeblurGAN an approach based on Conditional Generative Adversarial Networks, a multi-component loss function and a kernel-free blind motion deblurring learning approach. Different from previous works, Wasserstein GAN with Gradient Penalty and perceptual loss is used. It is a method based on random trajectories for generating a dataset for motion deblurring training in an automated fashion from the set of sharp images.

Xiangyu Xu et al. [5] present a deep convolutional neural network for extracting sharp edges from blurred images. The proposed model consists of two stages: suppressing extraneous details and enhancing sharp edges. The two-stage model simplifies the learning process and effectively restores sharp edges. It does not require any coarse-to-fine

strategy or edge selection, thus significantly simplifying kernel estimation and reducing computation load.

Zhao et al. [6] proposes a deep pyramid generative adversarial network with local and nonlocal similarity features, called LNL-PGAN which deals with motion image deblurring. They propose a nonlocal feature block, local feature block, and feature block. Thus the model can obtain nonlocal and local similarity features at multiple levels while capturing short range and long range dependencies to improve the network representation ability of the model. They also introduce the concept of multiscale generative adversarial loss which preserves the edge details and eases the sharp edge prediction.

Shuai Zheng et al. [7] propose an edge heuristic multi-scale generative adversarial network (GAN) , in the paper "Edge Heuristic GAN for Non-Uniform Blind Deblurring". It uses the coarse-to-fine scheme to restore clear images in an end-to-end manner. Edge-Generated Net is basically a lightweight GAN to generate heuristic edges. It contains a modified encoder-decoder network and ResBlocks to handle complicated motion blur. In Multi-Scale Deblurred Net a multi-scale network was designed as a GAN generator for image deblurring and the discriminator is parameter-shared for generators of different scales. The Loss consisted of Adversarial loss and Hierarchical loss.

Sungjun Lim et al.[8] present a novel unsupervised cycle-consistent generative adversarial network (CycleGAN) with a linear blur kernel, which can be used for both blind- and non-blind image deconvolution. It surpasses CNN approaches that usually require matched high resolution images for supervised training. The cycleGAN approaches existing prior to this had two CNN-based generators, whereas in this only a single CNN-based generator is necessary.

Boyu Lu et al [9] in their paper "UID-GAN: Unsupervised Image Deblurring via Disentangled Representations" presented an unsupervised method for single-image deblurring without paired training images. They proposed a disentangled framework which splitted the blurred image into original content of the image and the features of blur, which yielded better performance. The proposed framework consists of four parts: 1) content encoders for blurred and sharp image domains 2) blur encoder 3) blurred and sharp image generators 4) blurred and sharp image discriminators. During the testing time the blurring branch was removed, a blurred image was given as input from which the content and the blur features were extracted and finally the deblurred image was

generated by the generator.

Takuhiro Kaneko et al.[10] proposed the Noise Robust GANs which claim to learn clean image from the generator even when the input image is noisy. It can overcome the noisy input problem even without having the complete noise information which they achieved by introducing a noise generator and training it along with a clean image generator. The first model in the paper is based on the assumption that signal-noise relationship is priorly known. Multiplicative Gaussian noise and Poisson noise is considered and noise amount is trainable. In the next model, the first assumption still holds true along with which it is also assumed that the noise distribution type is known. In the third model an assumption similar to the second model is imposed but rotation and channel shuffle collapse the per-pixel signal-noise dependency that is included in typical signal dependent noise i.e. the assumption regarding color inversion is induced. The model is tested by comparing the results with various GAN models including AmbientGAN and the standard GAN.

Kaneko et al.[11] proposed noise-robust GAN (NR-GAN) which provides a partial solution to the above problem. NR-GAN has the ability to learn a clean image generator directly from noisy images. NR-GAN is capable of dealing with noise alone by assuming that information is lossless, both before and after the degradation. Irreversible degradation in the form of blur, compression or a combination can't be solved by NR-GAN. To address this issue, Kaneko et al. proposes blur, noise, and compression robust GAN (BNCR-GAN), which is capable of learning a clean image directly from blurred, noisy, and compressed images. It uses both blur robust GAN (BR-GAN) and compression robust GAN (CR-GAN) .

Computational time is another very important factor to keep in mind while dealing with image degradation. Tomosada et al.[12] in their paper propose a high quality image deblurring method that uses Discrete Cosine Transform (DCT) in order to reduce the computational complexity. Their innovation, DeblurDCTGAN preserves the texture and suppresses ringing artifacts in the restored image. It used an encoder-decoder architecture while also considering DCT based loss. By introducing a DCT generator loss in the generator, the ringing and blocking artifacts are reduced. As compared to other methods, this method has high deblurring performance while also suppressing excessive patterns.

Niu et al.[13] in the paper, "Blind Motion Deblurring Super-Resolution: When Dy-

namic Spatio-Temporal Learning Meets Static Image Understanding" recover sharp high-resolution images from motion blurred low resolution input. They model the reverse process of generation of blurred images. Three-streams network to recover the final sharp high-resolution images.

J. Dong and J. Pan [14] explore deep convolutional neural networks to directly estimate the confidence map in "Deep Outlier Handling for Image Deblurring - IEEE Transactions on Image Processing" . They identify reliable inliers and outliers and facilitate the deblurring process which can be applied to both non-blind and blind image deblurring.

Karaali et al. [15] in the paper,"Deep Multi-Scale Feature Learning for Defocus Blur Estimation" use CNN to tackle depth images from pattern edges. Multi-scale blur estimation is done for pattern edges that uses input patches with varying sizes.A fast edge-aware guided filter is used to propogate blur information estimated at pattern edge points.

In the paper, "Beyond Camera Motion Blur Removing: How to Handle Outliers in Deblurring", [16] a dataset synthesis approach is used that considers the outliers and consists of an edge aware scale recurrent net consisting of a deblurring part and an upsampling part. It has a salient edge selection network to supervise the deblurring process.

"Handling Outliers by Robust M-Estimation in Blind Image Deblurring" by Zhang et al. [17] proposes a blind motion deblurring method for blurred images including light streaks. It uses Huber's M-estimation method. Also, it solves the problems of false detection of candidate light streaks and optimal light streak in existing methods.

Zhao et al. [18] proposed in his paper, "Improved Deep Multi-Patch Hierarchical Network With Nested Module for Dynamic Scene Deblurring", nested module blocks which are substituted, whose powerful and complex representation ability is utilized to improve the deblurring performance. Attention mechanism is introduced to enable the network to differentiate blur across the whole blurry image from dynamic scene.

Subramanian et al. [19] in the paper, "SSIM Compliant Modeling Framework With Denoising and Deblurring Applications" , proposes a process of obtaining a generalised convex framework since SSIM has a non-convex nature making it difficult to be employed in model based applications like denoising, etc.

In the paper by "Image Denoising via CNNs: An Adversarial Approach" by Nithish Divakar et al. [3] an adversarial training based approach for denoising combined with

feature extraction layer was proposed. The feature extraction block used in this work is inspired by their paper.

1.6 Brief Methodology of the project

In order to execute the objectives of the project, three separate models can be taken and executed serially. Firstly, a Generative Adversarial Network(GAN) is used to deblur the image. This is followed by using a saturation handling network to take care of the saturation in the image.

2.2

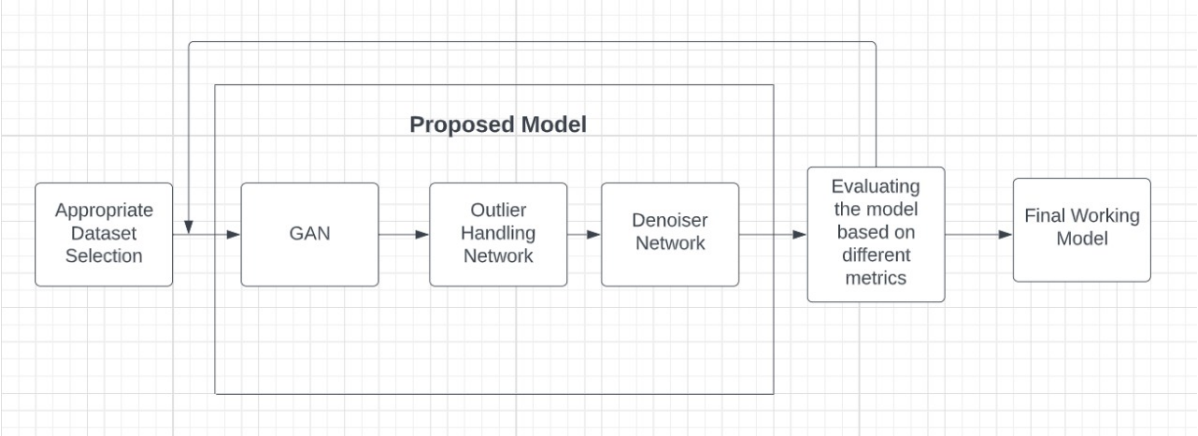

Figure 1.1: Proposed Methodology

Finally, an autoencoder is used to handle and remove the noise present in the image and generated during the entire process.

1.7 Assumptions made / Constraints of the project

The proposed model can be considered as a serial combination of three individual models.

The Generative Adversarial Network(GAN) model used for deblurring the image is run for 10 epochs, by training the model on 10000 images. The generated image can be much sharper if the model is trained on more images, and by running it for more epochs. The code for the architecture is also written using few inbuilt functions. If the code is written for these functions, the code will be more tailored towards generating a better output image depending on the input images.

While the saturation handling network can deal with the saturation in the images, it introduces a certain amount of noise since the architecture is a two-layered GAN structure

and GANs are susceptible to noise. While the fusion blocks in the structure are capable of getting rid of some amount of noise, they are not able to completely eliminate it.

When it comes to denoising network, it was assumed that only 4 types of noise namely gaussian,salt and pepper,speckle and impulsive noise will be present in the image after it is passed through the Deblurring and Saturation handling network.Therefore during training only these noises are synthetically added to sharp images and is used for training purpose.

1.8 Organization of the report

This report is organized as follows.

- Chapter 2 discusses the Theory and Fundamentals required for the implementation of Generative Adversarial Networks, saturation handling networks, and denoising networks(autoencoders). Each individual model is explained in brief, with the final subtopic giving a better insight into the working of the proposed model.
- Chapter 3 discusses the design of the proposed model. As the model is a serial implementation of three other models, the design of the three models is explained in brief. Next, the design of the combined, proposed model is also explained.
- Chapter 4 discusses the implementation of the proposed model. Again, as the proposed model is a serial implementation of the three other models, the implementation of the three models is discussed in brief before discussing the implementation of the combined model.
- Chapter 5 discusses the results obtained from the three models and the combined proposed model. The results have been compared with existing architectures in the same domain as well.
- Chapter 6 discusses the conclusion and the future scope of the proposed project. The learning outcomes of the project is discussed in brief as well.

Chapter 2

Theory and Fundamentals of Deblurring, Denoising and Saturation Handling

CHAPTER 2

THEORY AND FUNDAMENTALS OF DEBLURRING, DENOISING AND SATURATION HANDLING

Before taking a look at the design and implementation of the proposed model, it is essential to understand the basic working of the individual sub-models which work together to produce the desired output. This section focuses on understanding the theory and fundamentals of the sub-models, mainly the deblurring network, the saturation handling network, and the denoising network. Lastly, a theroetical understanding of the proposed model is also provided.

2.1 Deblurring

For deblurring of images, Generative Adversarial Networks(GAN) is used. More specifically, a variant of GAN known as SRGAN is used. SRGAN[18] is the type of Generative Adversarial Network(GAN) used for deburring and improving the resolution of an image. Here, SRGAN stands for Super Resolution GAN and is a very popular implementation method used for deblurring images. It is able to restore high-resolution image from low-resolution image by increasing the high-frequency components in the low-resolution image.

Before taking a look at SRGAN, it is important to understand the basic working of a GAN model. GAN treats the problem of image deblurring as a supervised learning problem. Two sub-models are used, the `generator model` that is trained to generate new examples, and a `discriminator model` that classifies the examples as real (from the domain) or fake (generated). The two models are trained together in an adversarial nature until the discriminator is fooled half the time, which indicates that the generator is able to generate plausible examples.

The generator is implemented first. The generator takes a fixed length random vector as input to generate a sample in the domain. Then, a vector is drawn randomly from the distribution and acts as a seed for the generative process. Points in this multidimensional vector space after training will match points in the problem domain. Latent variables are vectors found in the vector space (also known as the latent space). A data distribution is

projected or compressed by latent factors. Finally, new points extracted from the latent space may be fed into the generator model as input to produce fresh and novel examples of output.

The next sub-model is the discriminator. As mentioned, the discriminator predicts whether the images generated by the generator are fake or real. The generator model outputs generated examples.The generator is adaptable because it has mastered the art of accurately separating features from instances in the issue area. Utilizing the same or comparable input data, the feature extraction layers can be employed in transfer learning applications. A single value is produced by the discriminator's convolutional design. CNN or Residual Blocks can be used to model both the generator and discriminator.

Now that the sub-models are defined, the problem arises about making them work to achieve the same goal, which is to deblur an image successfully.

Together, the two models are trained. The discriminator is then updated to improve its ability to distinguish between genuine and false samples in the following round. Depending on how well the generated samples tricked the discriminator, or not, the generator is updated. The two models can be seen as being in a two-player or zero-sum game-like competition with one another. Zero-sum means that the discriminator is rewarded or no modification to the model parameters is required when it correctly distinguishes between real and false samples, whereas the generator is penalised with significant updates to the model parameters. Similar to this, when the generator deceives the discriminator, it gets rewarded or the model parameters do not need to be changed, but the discriminator suffers consequences and has to update its model parameters.

Coming to the uses of GAN networks, it can be used to handle high-dimensional data, missing data, and provide numerous plausible responses or multi-modal outputs. It can also be utilised for further and perhaps more domain-specific methods of data augmentation. GANs are most effectively used in conditional GANs for tasks requiring the creation of additional instances. Another use is for Super-Resolution of images where it creates high-resolution copies of the input photos. Furthermore, GAN has the capacity to produce original, artistic drawings, paintings, and other things. Also, it has the capacity to convert images between different climates, including summer and winter and other seasons.

The architecture of SRGAN is the same as mentioned above. The different variants of

GAN use different architectures for designing and implementing both the generator and the discriminator. The design of the generator and discriminator is very well explained in the coming chapter. A simple illustration of the generator and the discriminator block is provided here.

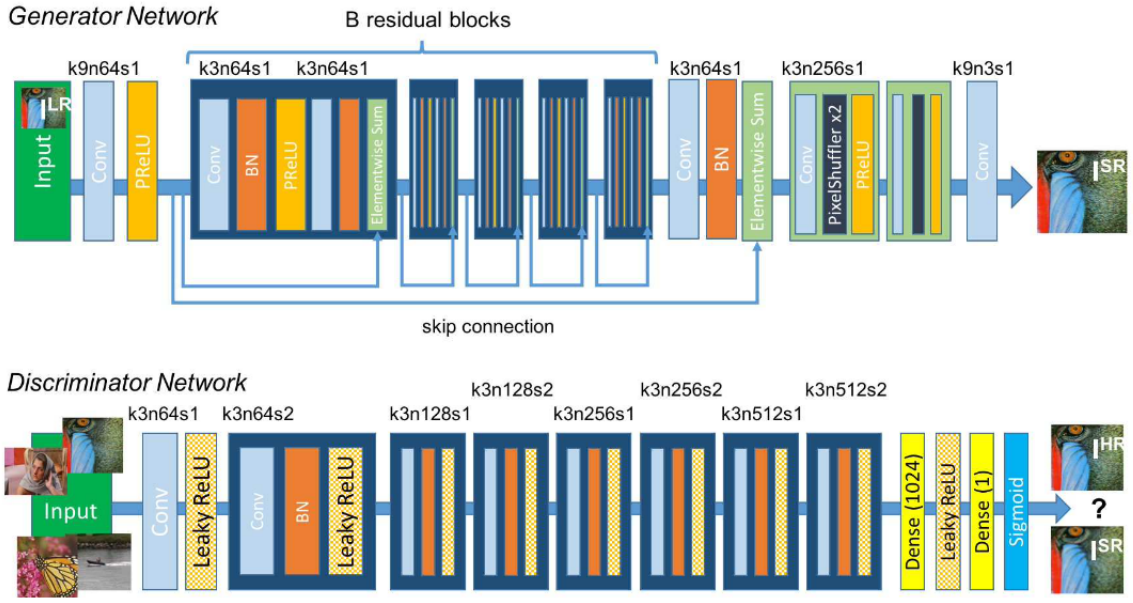

Figure 2.1: Architecture of SRGAN

Coming to the drawbacks with SRGAN, it is seen that the model struggles when it comes to dealing with textures in the image. Texture often appears distorted and deformed in the image which is generated by the network.

2.2 Saturation Handling

The saturation handling module in our project has up-sampling and fusion subnets. ESRGAN (Enhanced Super Resolution GAN) [**2**] which is a variant of GAN was modified to realise this. A GAN model was chosen for saturation handling since the deblurring module is using GAN too, making the saturation handling module more compatible with the previous module.

The reason ESRGAN was chosen is that it performs better than other super resolution methods due to the following reasons:

1. Batch Normalisation Layers are removed

2. Using features before activation helps generator recover more realistic texture details

3. Network interpolation of ESRGAN and PSNR model for removing unpleasant noise while maintaining good perceptual quality

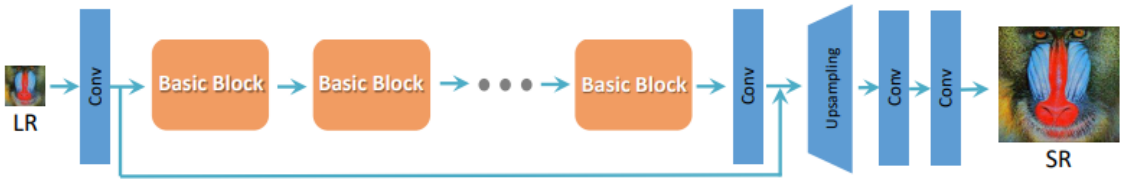

Figure 2.2: Architecture of Regular ESRGAN

The saturation module acquires high frequency information from the up-sampled image and low frequency components from the original image and fuses them to average out the saturated pixels. In this project, ESRGAN was modified to layer it in such a way that there are 2 upsampling subnets and 3 fusion subnets because it was observed that layering it in this way produced the best results.

2.3 Denoising

Image denoising is one among the fundamental problems when it comes to image processing. Objective of the image denoising is to remove the noise present in the image while preserving the original image structure. Traditionally, filtering and wavelet transforms have been the mainstay image denoising methods. Recently, machine learning has become a new approach applied to denoising digital images.In machine learning Neural Networks(NNs) were first used for denoising images.They had fully connected layers which were few layers deep. Stacked auto-encoders (SARs) and deep belief networks (DBNs) were the typical NNs for denoising introduced initially. They used stacked layers in an unsupervised manner to train the models and obtain good performance. However, these networks were complex and also required too many manual settings in order to produce optimum results. Although the majority of deep learning methods for denoising have produced acceptable results, they have a number of flaws, such as the necessity for test phase optimization methods, manual parameter setup, and a specific model for single denoising jobs.

Due to this, end-to-end connected networks, especially Convolutional Neural Networks(CNNs), were proposed. CNNs find applications in the wide range of fields of image processing.Since CNNs produced decent results with other image processing prob-

lems they were also trained for denoising images.CNN-based methods such as DnCNN[20] and FFDNet [21] gave promising results also.

A learning architecture known as an autoencoder [22] develops output images that are highly similar to the input images. This architecture compresses the information contained in the input image and lowers the dimensionality of the image data by learning identity mapping. On the basis of noisy input photos, the Denoising Autoencoder, which is based on the Autoencoder learning architecture, trains to provide clean forecast images. Even if the analysis subjects undergo minor, irrelevant changes, the Denoising Autoencoder seeks to uncover intriguing structure in the input distribution. Architecture of autoencoders involving fully connected layers is shown in 2.3

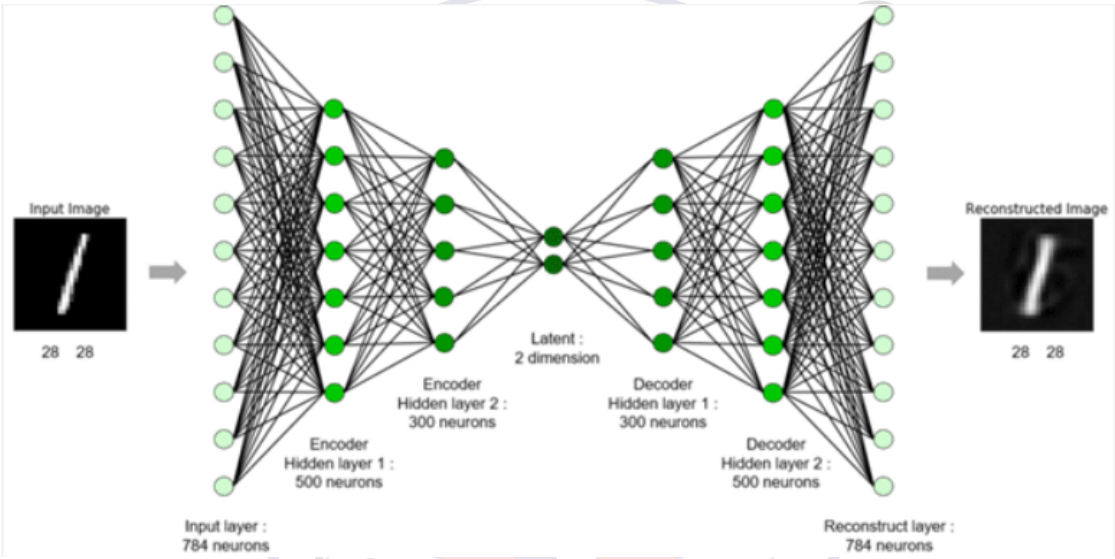

Figure 2.3: Architecture of Autoencoders with Fully-connected layers

Keeping in mind the success of CNNs the Convolutional Autoencoders were introduced for denoising images. Architecture of Convolutional Autoencoders is show in the figure 2.4. An important tradeoff in autoencoders is the bias-variance tradeoff. On the one hand, we want the architecure of the autoencoder to be able to reconstruct the input well (i.e. reduce the reconstruction error). On the other hand, we want the low representation to generalize to a meaningful one [23]. This property of autoencoders can also be leveraged for the denoising applications. In the Denoising CNNs, CNN is used for image noise reduction. Unlike conventional autoencoders with fully connected layers, CNN is used for encoding and decoding parts of an autoencoder.

By adding more layers, one would anticipate that the CNN model would excel at the

denoising task. Unfortunately, the deeper CNN model does not always produce superior results; occasionally, the deeper model produces worse outcomes than the shallower one. According to this method, the deeper layers in the model must learn the identity mapping in order to prevent changes from being added after the shallower layers have acquired sufficient knowledge.The layers struggle with this approximate task. Because of this, trained deeper models run into the so-called degradation problem and are unable to produce results that are satisfactory. Apart from this deeper CNN models also suffer from gradient explosion which significantly inhibits the learning capacity of the network.

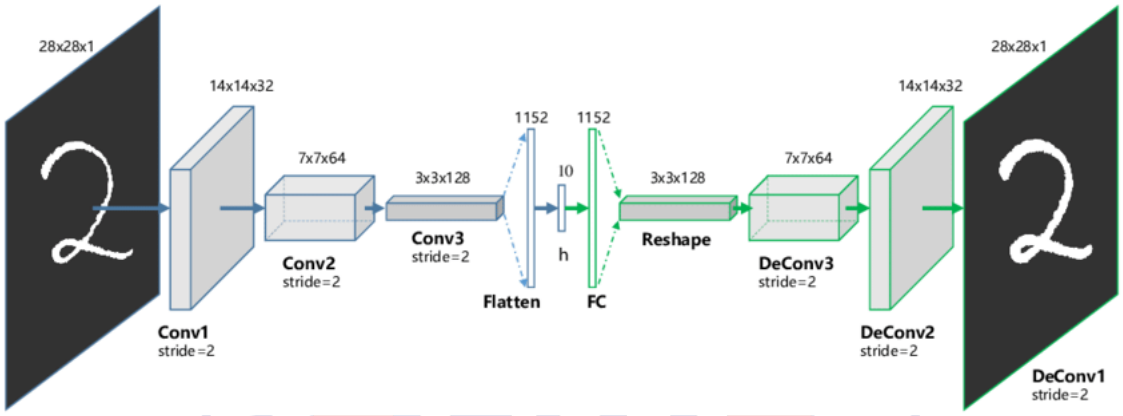

Figure 2.4: Architecture of Convolutional Autoencoders

This project overcomes the mentioned shortcomings of DCNNs by implementing:

1. skip connections to overcome the problem of gradient explosion and gradient degradation
2. Using features extraction layer to recover more realistic texture details
3. Training the model on different types of noises namely gaussian noise,speckle noise and impulsive noise

The implementation details of the proposed model is discussed in coming sections of this report.

2.4 Tools Used

1. Frameworks: PyTorch, Tensorflow, Keras
2. Software: Google Colab, Visual Studio
3. Hardware: The GeForce® GTX 1650

2.5 Summary

This chapter briefly explains the basic models that are used to build the proposed model. By understanding the fundamentals of the GAN architecture, the implementation code is written for the same. By exploring the uses a better understanding of how to correctly implement the model is obtained. The shortcomings of this model are also looked into. Based on the drawbacks of existing methods and the requirements of this project (like time complexity and compatibility with the other modules in this project) are taken into consideration while finalising on ESR-GAN as the base for the Saturation Handling module. Moreover, the history of denoising, how it evolved, and the shortcomings of the traditional approach is explained. Also, how those shortcomings are overcome in this project is mentioned. In the coming chapters, lets dwell deep into the design and implementation of the modifications to the existing architectures to obtain better performance.

Chapter 3

Design of End-to-End Deblurring with Denoising and Saturation Handling Architecture

CHAPTER 3

DESIGN OF END-TO-END DEBLURRING WITH DENOISING AND SATURATION HANDLING ARCHITECTURE

In this section, a deeper insight is provided into the design of the proposed model. In line with the rest of the report, we take a look at the design of the three individual sub models. Apart from a brief look at the design, the batch size, epochs and various other parameters in the proposed model are also discussed.

3.1 Models used

The three main sub models used are the deblurring network, the saturation handling network, and the denoising network.

3.1.1 Deblurring Network (SRGAN)

The implementation of SRGAN is done using the help of Google Colab. Using the help of the mount function in colab, the MIRFLICKR dataset is used for implementation. All the necessary packages are imported as well.

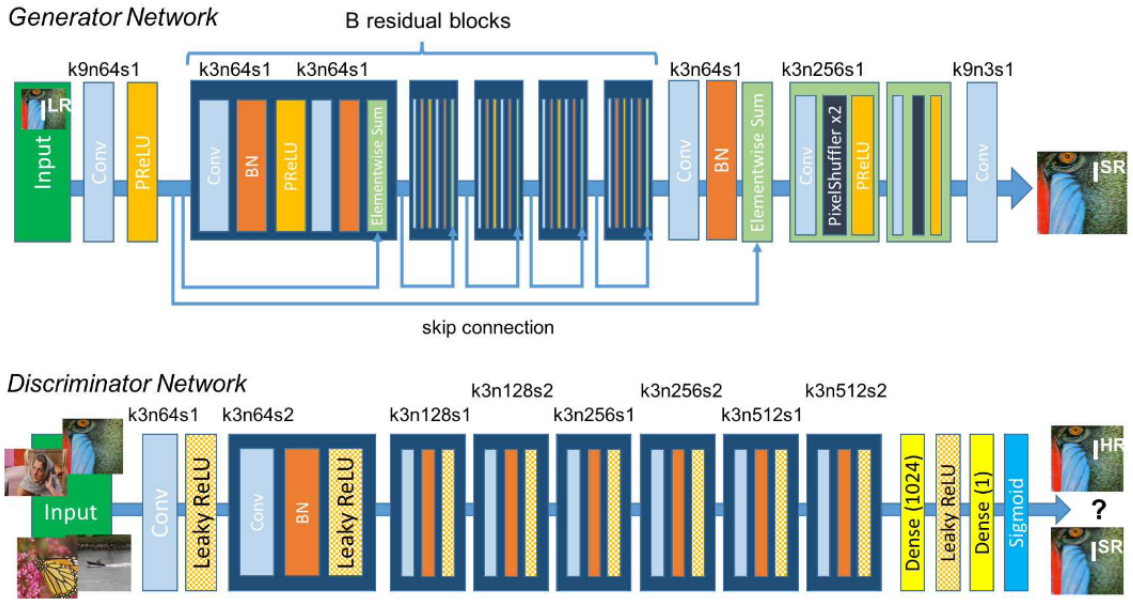

Figure 3.1: Deblurring Network (SRGAN)

As explained earlier, low-resolution and high-resolution images are needed. So, the images from the dataset are converted into both low-resolution and high-resolution images. The image is rescaled while retaining the aspect ratio as required. To obtain low-resolution images, the original images are resized to dimensions of 32 x 32. To obtain high-resolution images, the original images are resized to dimensions of 128 x 128.

The following blocks are made in accordance to the figure 3.1 . These are the main blocks which are used to build the SRGAN sub-model.

Coming to the generator, it can be implemented using the help of either CNN or Resnet Blocks. Here Resnet Blocks are used as the deep CNN faces the problem of degradation. As the depth of the model increases, the accuracy starts to saturate. Resnet block consists of 2D convolution layer, followed by batch normalization and PReLU (Parametric Rectified Linear Unit). All of them are inbuilt functions. For 2D convolution, it learns from 64 filters, and kernel size is 3x3.

Upscale Blocks are also used. Similar to Resnet block, the upscaling block has batch normalization replaced with upsampling function.

Next part is the main generator model. Here, 16 Resnet blocks are used. These blocks create a new image, upscale it. The model returns both the input image and the new generated image.

Next is the discriminator block. This block is similar to a Resnet block. Instead of PReLu, it uses Leaky ReLu. The slope of PReLu is determined by the Resnet block, but it is fixed in Leaky ReLu.

The discriminator model makes use of the discriminator block. The model alternates between stride 1 and stride 2. Stride specifies the step of the convolution. Stride modifies the movement over the image. Higher stride value usually makes the output size smaller.

The next block used is the combined model. Both, low resolution and high resolution images are given as the input to the model. It gives two outputs, namely Validity and VGG 19. Perceptual loss function used is a mixture of content loss and adversarial loss. Content loss has a higher weight. Validity is the adversarial loss, i.e. whether the discriminator can correctly make the prediction or not. VGG 19 is the content loss. VGG converts pixel space into feature space. VGG loss is the Euclidean distance between the feature representation of the reconstructed image. The weights help the model know the features to look at. The Top False parameter lets us include any input shape. Otherwise,

the default shape considered is 224x224. Output is layer 10, the last convolutional layer. In short, VGG compares the features from the actual image with the generated image.

3.1.2 Saturation Handling Network

Implementation of the Saturation Handling module was on Visual Studio. A layered structure of ESRGAN was developed for upsampling and fusion blocks were added which fused the low resolution input image with the high resolution output image. This combined ESRGAN and fusion block structure was layered since that was handling saturated pixels better than a single Upsampling + Fusion block with the same total scaling power.

Other possible ways of upsampling include using an Encoder-Decoder structure instead of a GAN structure but the GAN structure was finally implemented as it was seen to give better results and was more compatible with the previous deblurring module which uses SRGAN.

During fusion, the upsampled image was given higher weightage than the input image in the first layer but both were given equal weightage in the second layer. In the first layer, the weight given to the high resolution image was 0.6 and the low resolution image was given 0.4. In the second layer, both high resolution and low resolution images were given 0.5 weightage. In the third and final fusion block, both the images are given equal weights, i.e, 0.5 weightage each.

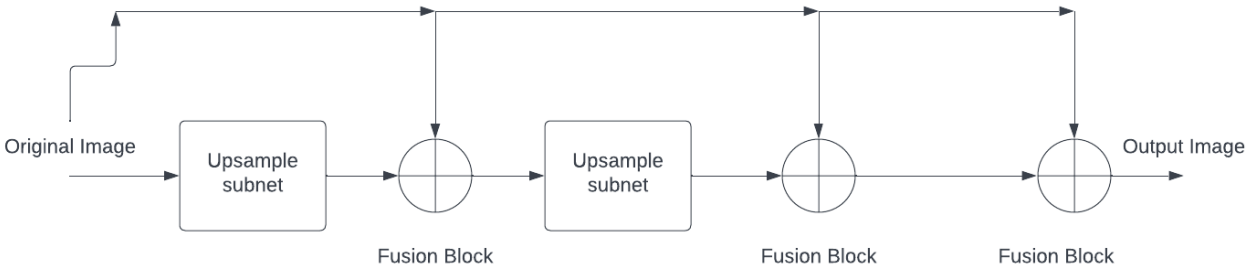

Figure 3.2: Saturation Handling Network

It was observed that the first two layers were introducing some amount of noise to the images as GANs are susceptible to noise because of which the third fusion block was added as it was seen to reduce noise.

3.1.3 Denoising Network

The Denoising network is written using tensorflow framework. Flickr image dataset from the kaggle is used for training the network. Among the 32785 images 2000 images is chosen randomly and is used for training and validation with validation split = 0.1.The images were reshaped to 128x128 and is used for training and validation.During the preprocessing of images different types of noise such as gaussian noise,speckle noise and impulsive noise is added.This makes the training set diverse.

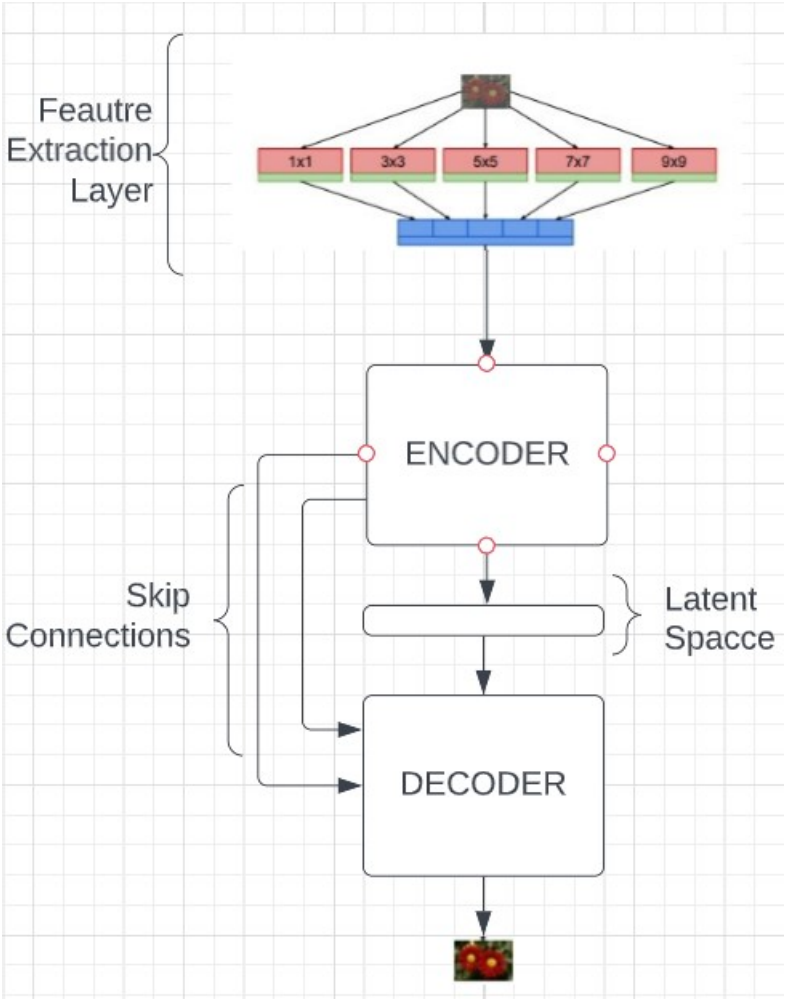

Figure 3.3: Architecture of Denoiser Network

The model is derived from the conventional CNN based Autoencoder network.The encoder part is 8 layer deep which is then connected to fully connected layer which serves as latent space.Latent space is fully connected to the decoder part which acts as transpose of encoder block which employs deconvolution blocks. To this two skip connections were added across the encoder and the decoder part.

Filters sizes decreases from 5x5 to 3x3 to 1x1 as the depth of the layers increase in

the encoder. Reverse is true for decoder. This is done to preserve the edge details of the image and to prevent over-smoothening.Strides of 2x2 is used for both encoders and decoders.

Instead of directly feeding the image as input to the network the image is fed to feature extraction part which contains filters of sizes varying from 1x1, 3x3, 5x5, 7x7 and 9x9.These extracted features are stacked upon one another and is fed to encoder-decoder block for denoising. Binary cross entropy is used as the loss function.

3.2 Specifications for the Design

In this subsection, a deeper insight is provided into the specifications of each of the individual sub models.

3.2.1 Deblurring Network (SRGAN)

As mentioned earlier, blocks of code are written for the blocks that are used in the generator and the discriminator. The Resnet block uses a convolutional kernel size of 3x3, which has 64 filters. The same number of filters and the same kernel size in used in the generator model block. Batch normalization has the momentum of the moving average set to 0.5 in the both the blocks of code. However, the upscaling block has 256 filters, while retaining the same 3x3 kernel size. Here, the upsampling layer has size set to 2, indicating that 2 tuples of integers are used. The generator model uses three convolutional layers, two of which have filter number as 64 and kernel size as 3x3. The last convolutional layer has 3 filters while having a kernel size of 9x9.

On the other hand, the discriminator takes the number of filters as an input. The kernel size remains 3x3. The momentum of batch normalization is 0.8. The slope for LeakyReLu activation is set to 0.2.

In contrast to typical GAN architectures, here, discriminator.trainable is set to false. This implies that the discriminator is not being trained. Only the generator model is updated depending on the output of the discriminator. Doing so reduces both the time and space complexity considerably.

3.2.2 Saturation Handling Network

The Residual Dense Block has a growth channel of size 32 and convolutional kernel of size 3x3. In the RRDBNet module, the upscaling takes place twice and the scaling factor used each time is 1.45 with a LeakyReLu negative slope of 0.2. The final Residual Dense

Block Network that layers multiple Residual Dense Blocks has convolutional kernel of size 3x3 and filter size of 64.

In the first fusion block introduced to ESRGAN, the weight given to high resolution image was 0.6 and low resolution image was given 0.4. In the second fusion block, both high resolution and low resolution images were given 0.5 weightage. In the third fusion block, both high resolution and low resolution images were given 0.5 weightage.

3.2.3 Denoising Network

For denoising, instead of directly passing the image to autoencoder network it was fed to a feature extraction block. The feature extraction block consists of filters of size 1x1,3x3,5x5,7x7 and 9x9 and has 8,16,24,32 and 40 filters respectively. LeakyRelu activation is applied on the resulting convolution operations. These activations are stacked together to form the Feature Extraction Layer.

The Feature extracted from the previous layer is fed to the autoencoder network.The filter sizes of the autoencoder network progressively decreases from 5x5 to 3x3 and then to 1x1. This is done to prevent the over smoothening of the image as well to lighten the model. Skip connections are given from 3rd and 5th layer encoder block to the corresponding decoder layers.

3.3 Design Methodology

3.3.1 Deblurring Network (SRGAN)

An extensive literature survey was done to find the best GAN architecture for the proposed model. Apart from SRGAN, few of the architectures considered were CycleGAN, DeblurGAN, Edge Heuristic GAN, and Noise Robust GAN. CycleGAN is unsupervised with a linear blur kernel while DeblurGAN treats the entire problem as image-to-image translation. Edge Heuristic GAN is more focussed towards handling motion blur, while Noise Robust GAN gave a lot of emphasis on removing the noise in the input image. Super Resolution GAN on the other hand focusses solely on the problem of generating a high resolution image from a low resolution image, which is in line with the problem statement.

3.3.2 Saturation Handling

After going through multiple Saturation Handling approaches, it was seen that use of Confidence maps (using CNN), Robust GANs and ResNet Encoder-Decoder structures are some of the viable methods. In the confidence map method, outliers and inliers are identified and the contribution of outliers is excluded while generating the final image but in this method, a certain amount of information is lost since components from outlier pixels is completely ignored. To avoid this, it was finalised to perform upsampling of the low resolution image and eventually fuse the low resolution and high resolution images to average out the saturated pixels. This way the information from all pixels are taken into consideration. For upsampling ESRGAN was chosen over Encoder-Decoder structures since GANs are seen to have slightly better performace for upsampling than Encoder-Decoder structures and also because it is more compatible with the GAN architecture used in the previous module of this project whose output is passed into this module.

3.3.3 Denoising Network

After going through the existing denoising architectures,it was decided that it is better to go for traditional denoising methods such as Denoising Autoencoders.These networks outperform learning-based methods when evaluated with real images.

Among the denoising autoencoders through analysis it was found that Denoising network with the skip connections produced better results in terms of visual quality hence we can say that the performance of the model is improved. “Binary cross entropy” as a loss function results in smoother convergence of the validation and training loss compared to the “mse” loss function. A feature extraction layer is added which consists of filters of various sizes to optimally extract features from the images which can be used as input for autoencoder instead of directly giving the image itself.

Our proposed model aims to use denoising networks to build a realistic noise model and combine it with an optimal GAN model to deblur and denoise image.

3.3.4 Proposed Model

The proposed model is a serial implementation of the models mentioned in the above subsection.

Existing deblurring techniques do not deal with outlier handling and denoising. GAN has been known to produce better results with respect to deblurring as compared with

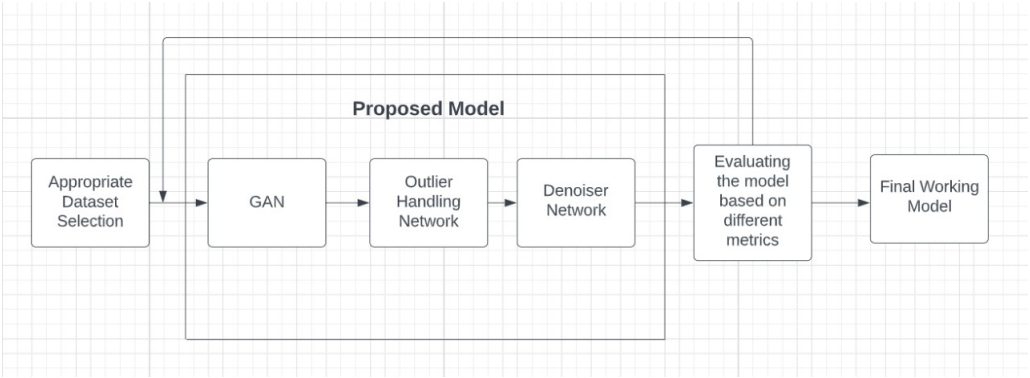

Figure 3.4: Proposed Model

other existing CNN models. This study deals with realizing an optimal GAN model that can handle outliers as well as denoising, which is better than state of the art models.

3.4 Summary

So this chapter gives a brief look into the designing part of the three sub models used in the proposed model. With the help of a deep understanding of the models, the constraints, and the design specifications, the implementation of the proposed model can be explored.

Chapter 4

Implementation of End-to-End Deblurring with Denoising and Saturation Handling

CHAPTER 4

IMPLEMENTATION OF END-TO-END DEBLURRING WITH DENOISING AND SATURATION HANDLING

This chapter focuses on the implementation of all the sub models, which mostly focuses on how the training and testing is carried out. Like in the previous chapters, the implementation of the individual sub-models is first explained before taking a look at the implementation of the proposed model.

4.1 Implementation of Deblurring Network (SRGAN)

The implementation of the SRGAN architecture is done on Google Colab.

4.1.1 Frameworks used for implementation

The implementation code of the SRGAN architecture is made in accordance with the design of the individual sub blocks as mentioned in the previous chapter. The keras library is used for the implementation of the convolutional layers, batch normalization, and the other basic layers used in the individual sub blocks. Tensorflow is another framework used for implementing machine learning concepts. The sklearn library is used for splitting the images into training and testing datasets. The split is 67% for training and the remaining 33% for testing.

After the implementation of the code for the individual blocks, the code for the combined model (generator and discriminator) is written.

4.1.2 Training

The batch size is taken as 1. Two arrays are created, one for storing the training low resolution images and the other for storing the training high resolution images.

Next, the code is made to run for 10 epochs. For every epoch, a label of 0 is assigned to all the fake images that are generated, while a label of 1 is assigned to all the real images. Furthermore, two other arrays are created for storing the discriminator and generator losses. Following this step, the fake images are created. Then, the discriminator is trained on both the fake and real high resolution images.

The next step is training the generator by fixing the discriminator as non-trainable.

After this step, the discriminator loss is averaged, for reporting the losses. Then, the VGG feature extraction is done for calculating the loss. Then, the generator is trained via the help of GAN. The losses used for training are adversarial and content loss. The losses are then stored in the two arrays as mentioned before. The list of arrays is converted to an array to make it easy to average the losses. Then, the average losses are calculated for both the generator and the discriminator. After every 5 epochs, the model is saved, which is then used later for generating the super-resolution image.

4.1.3 Testing

The last step in this sub model involves loading the model saved before. Once the model, is loaded a random image pair is taken, and the super-resolution image is generated. The generated image is compared with the low-resolution image and the PSNR value is calculated.

4.2 Implementation of Saturation Handling Network

The implementation of the Saturation Handling Network is done on Visual Studio using PyTorch. It consists of two upsampling layers and three fusion blocks.

4.2.1 Up-sampling Block

First a Residual Dense Block is implemented which in turn is used for implementing the Residual-in-Residual Dense Block (RRDB) by cascading the Residual Dense Blocks.

These Residual-in-Residual Dense Blocks are then layered to form the Residual-in-Residual Dense Network (RRDBNet) and are given a kernel size of 3x3. The LeakyReLU layer in this block is given a negative slope of 0.2 and up-convolution is done in the network twice with a scaling factor of 1.45 both the times.

The output of this is the first ESRGAN layer output.

4.2.2 Fusion Block

After the first up-sampling bock, a fusion block is added and this block fuses the first up-sampled output with the original image with weights 0.6 and 0.4 given to the high resolution and low resolution image respectively.

4.2.3 Combined Structure

Then a second layer of the complete Up-sampling + Fusion Block structure is added. The input to this layer is the first fusion block's output. This output is then passed into

the second fusion block where it fuses this output image with the original image with 0.5 and 0.5.

The output from the second fusion block is then passed into the third fusion block for reducing noise and this again fuses the previous output with the original image. The weights used in this fusion block are 0.5 and 0.5 again.

This complete process generates an image with reduced saturation.

4.3 Implementation of Denoising Network

Denoising network used here is a combination of autoencoder with skip connections and feature extraction layer. This is implemented in accordance with the details mentioned in the previous subsection. Implementation of the convolutional layers, batch normalization, and the other basic layers used in both feature extraction block as well as autoencoder block is done using the Keras library. Tensorflow is another framework used for implementing machine learning concepts.

4.3.1 Feature Extraction Block

The feature extraction block consists of filters of size 1x1,3x3,5x5,7x7 and 9x9 and has 8,16,24,32 and 40 filters respectively. LeakyRelu activation is applied on the resulting convolution operations. These activations are stacked together to form the Feature Extraction Layer. Strides of 1x1 is used in feature extraction block.

4.3.2 Autoencoder

The features extracted from previous layer is fed to convolutional autoencoder block. Auto encoder block consists of filters of various sizes. The filter sizes decreases progressively from 5x5 to 1x1 as we move to the deeper layers. LeakyReLu is used as activation function. In the encoder, for the first two convolutional layers the filter size is 5x5 followed by three convolution layers with the 3x3 and then two more convolutional blocks of 1x1. The obtained features are reshaped and connected to the latent space using dense layers. Strides of 2x2 is maintained throughout the autoencoder block.

Both the feature extraction and autoencoder block is stacked serially and trained. Training images consisted of images from FLICKR 8K dataset. This contained both indoor as well as outdoor images. To these images synthetic noise was added using the defining functions of different types of noises namely gaussian noise, salt and pepper

noise, speckle noise and poisson noise. These noisy images are paired with their sharp images and is passed for the training. 2000 images are used for training and validation of the network. Validation split was kept as 0.9. Batch size of 4 is used for training. The model is trained for 60 epochs.

4.4 Summary

Extensive details of the implementation of each sub model is provided in this chapter. The individual networks are arranged serially in order to get the final output. Training and optimization of the individual sub models is also carried out. The order of arrangement of the model goes like Deblurring network-Saturation Handling network-Denoising network.

Chapter 5

Results & Discussions

CHAPTER 5

RESULTS & DISCUSSIONS

In this chapter, we explore the working of our model on the popular image dataset MIRFLICKR. As we can see, the proposed model is able to deblur the input image, handle the saturation, and denoise the image to a great extent. We compare the output of the proposed model with the original ground truth image.

5.1 Simulation Results

In this section, we first look at the performance of the individual models on the popular Lena image. After analyzing the performance of each submodel, we take three pictures and pass it through the proposed model to get the final deblurred and denoised image where saturation is also handled.

5.1.1 Deblurring Network (SRGAN) Simulation Results

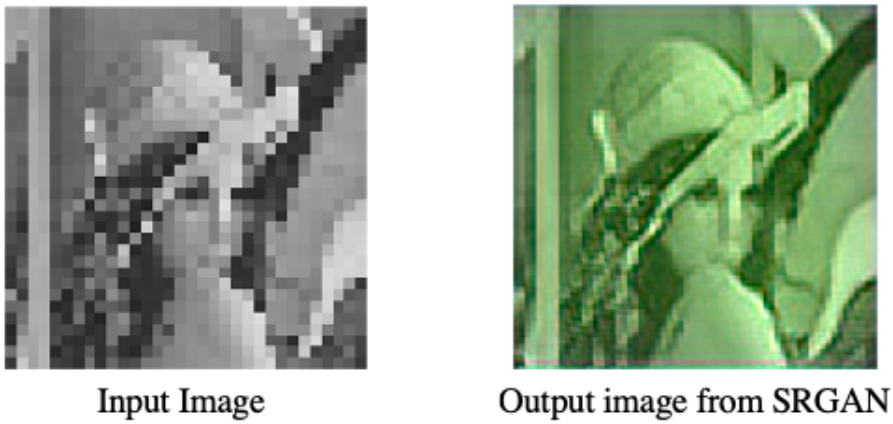

Figure 5.1: Comparison between Input image (size:32x32) and Output image from SRGAN (size:128x128)

Here, the SRGAN model takes the Lena image as the input. The model is trained for 10 epochs. In 5.1, the image on the right is the deblurred output of the image fed to the SRGAN model.

5.1.2 Saturation Handling Network (ESRGAN) Simulation Results

Figure 5.2: Unsaturated and Saturated Image

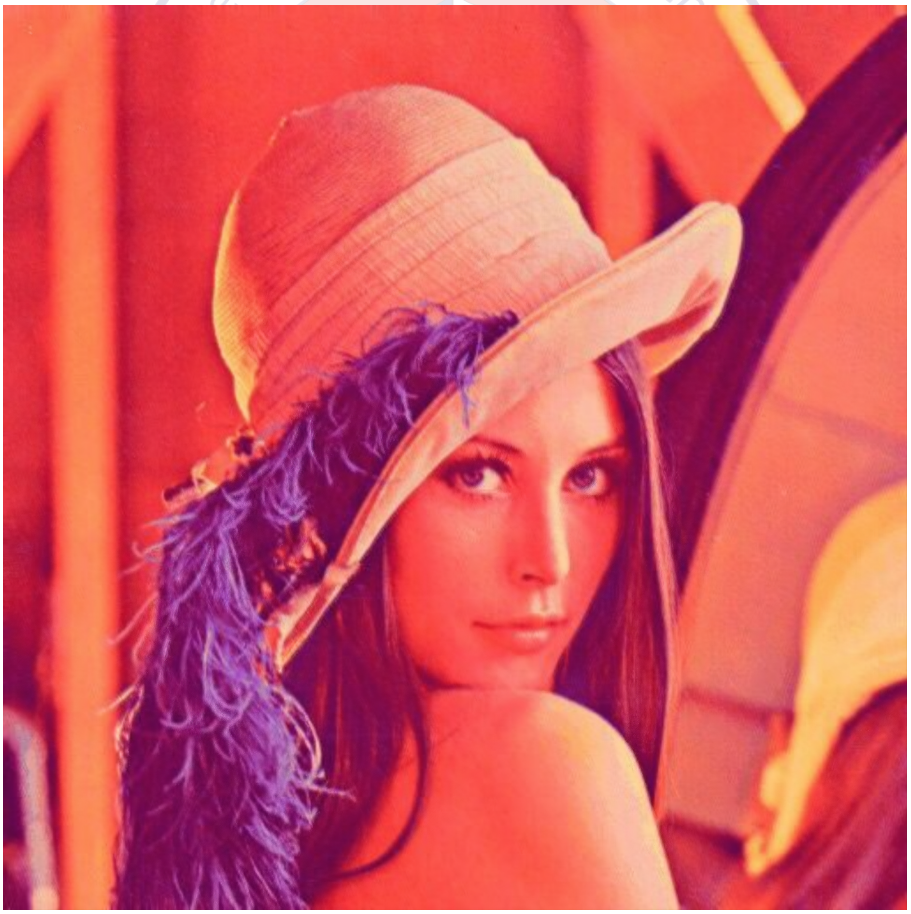

Figure 5.3: Output Unsaturated Image

Here, the original Lena image is taken and saturation is added to the image, depicted in 5.2. The image is then passed through the ESRGAN model to get the unsaturated

output. Even though the output is not equivalent to the original, unsaturated image, it can be seen that it has considerably less saturation as compared to the saturated input image. The output from ESRGAN (5.3) is also upscaled, as a result of which the output image is also much bigger than the input image.

5.1.3 Denoising Simulation Results

The famous Lena image is taken and random noise is added to it using the inbuilt python functions. This noisy image is passed to the model as input to measure the performance of the model with different configurations. Model is trained for 30 epochs in every case.

Figure 5.4: Sharp and Noisy Image

Figure 5.5: Output of the denoising network without skip connections and without feature extraction layer

Figure 5.6: Output of the denoising network when input image was fed directly without feature extraction layer

Figure 5.7: Output of the denoising network when input image was fed to feature extraction layer

Above are the output of the denoising model with different configurations when the noisy image from 5.4 was fed as input. Visually both the outputs 5.6 and 5.7 appear same but when psnr value is calculated the superiority of the one over the other can be verified. 5.6 shows the psnr values. It can be observed that there is no image in 5.5 this is because the model without skip connection takes larger number of epochs to learn that too with very less efficiency because of exploding and vanishing gradient problem.

5.1.4 Proposed Model Simulation Results

Figure 5.8: Comparison between Input image,Output after every stages and the Ground truth

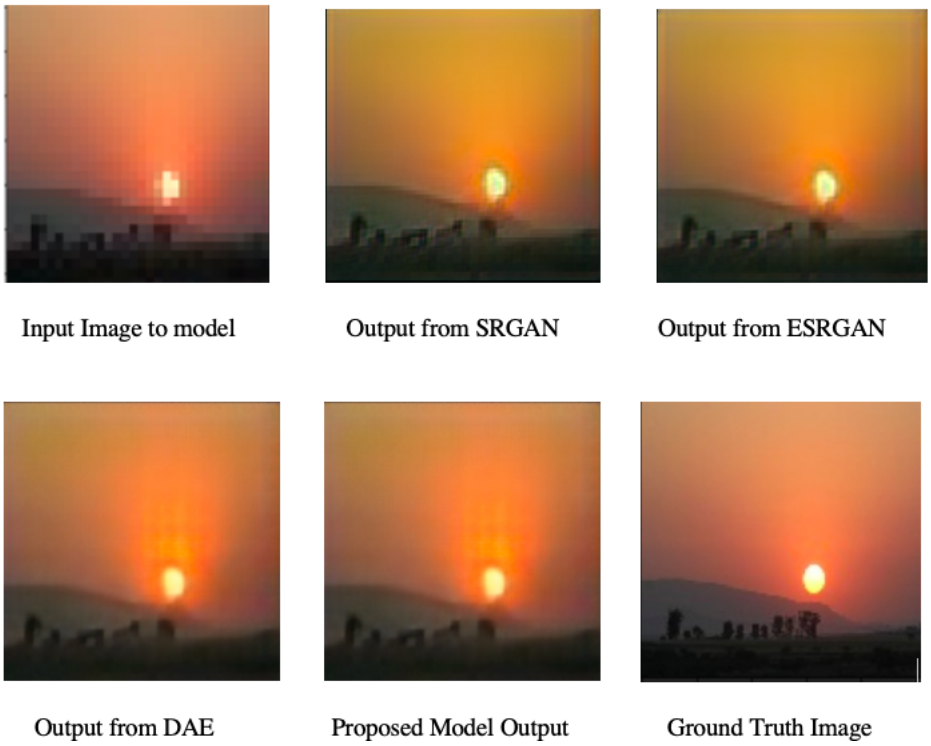

Figure 5.9: Comparison between Input image,Output after every stages and the Ground truth

Figure 5.10: Comparison between Input image,Output after every stages and the Ground truth

This section shows the simulation results of the entire project. The proposed model is run for three different images. The images on the left hand side, i.e. the blurred image with noise and saturation is given as the input to the model. The images in the middle are the output from the proposed model. These images are deblurred and denoised. Also, the saturation from the input images is removed in these images. Lastly, the images on the right are the original ground truth images, which gives a comparison with the output from the proposed model.

5.2 Experimental results

5.2.1 SRGAN generator and discriminator losses

An important aspect of SRGAN is reducing the generator and discriminator losses after every epoch. The following image shows the generator and discriminator losses after the SRGAN code is run for 10 epochs. Here, it is seen that as the number of epochs increases, both the generator and discriminator losses reduce.

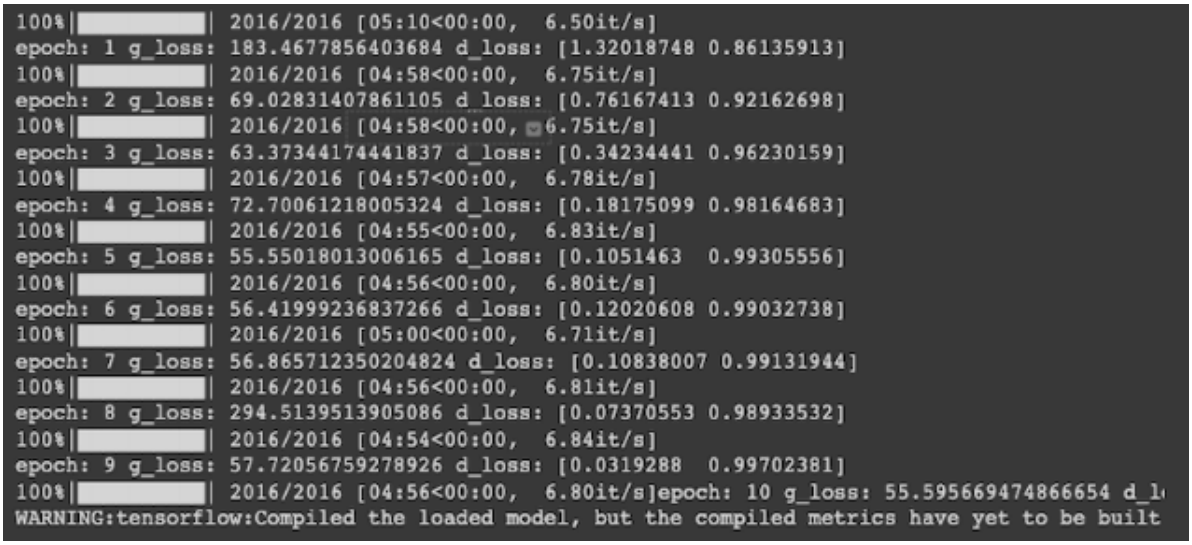

Figure 5.11: Generator and discriminator losses for SRGAN

Table 5.1: Comparison of SRGAN parameters before and after modification

Parameters	**SRGAN**	
	Original	**After Modification**
Total Parameters	38,249,281	42,619,140
Trainable Parameters	1,003,712	2,040,067
Non-Trainable Parameters	37,245,569	40,579,073

In 5.1, a deeper insight is provided into the total number of parameters, number of trainable parameters, and the number of non-trainable parameters, for both the original and modified SRGAN architecture.

5.2.2 Saturation Handling Network

While saturation in images can be recognised through the visual quality of images, PSNR and SSIM values were also calculated at every stage to test the performance of the model in other aspects with each modification.

Table 5.2: PSNR and SSIM values of each Individual Model and Proposed Model

Metrics	**Individual Models**		
	Original Model	**Intermediate Model**	**Final Model**
PSNR (image 1)	29.75	29.83	29.96
SSIM (image 1)	0.57	0.65	0.70
PSNR (image 2)	35.03	35.54	36.21
SSIM (image 2)	0.92	0.94	0.96
PSNR (image 3)	33.12	33.21	33.40
SSIM (image 3)	0.68	0.69	0.75

As seen in 5.2, the first column depicts the performance metrics of the original ESRGAN model. The middle column depicts the metrics of the intermediate model with the double-layer of ESRGAN and addition of a fusion block after each layer. The right column depicts the metrics of the final model which consists of three fusion blocks.

As can be seen from the metrics, the layering of ESRGAN and the addition of fusion blocks has improved performance of the model while also handling saturation.

5.2.3 Denoising Network

Before coming to conclusion about the inclusion of skip connections in the model a comparitive study is conducted to know how the model performs with and without skip connections.

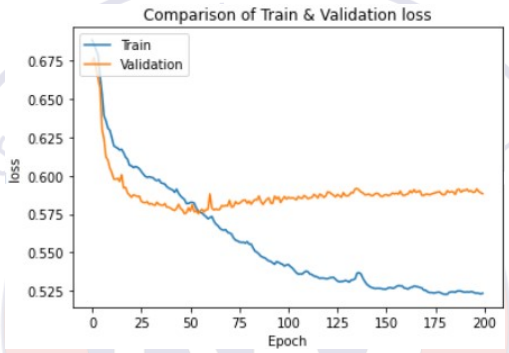

Figure 5.12: Training and validation loss without skip connections

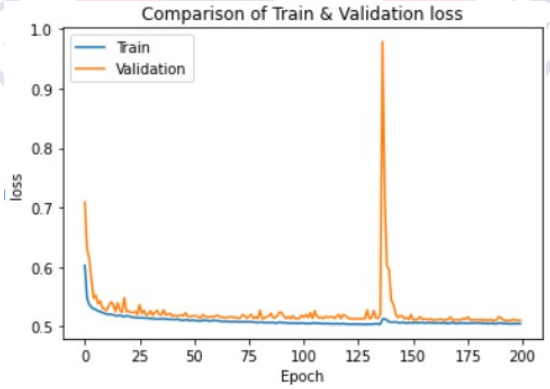

Figure 5.13: Training and validation loss with skip connections

Its clear from both images that the model with skip connections performance is superior than the one without the skip connection. This can be attributed to the fact that skip connections help in overcoming the problem of vanishing and exploding gradients. Further increase in number of skip did not bring any improvement to the performance of the model so only two connections were maintained.

When it comes to losses there was a need to choose among "mse" and "Binary cross entropy" loss function. This selection was also done on the basis of experimental results.

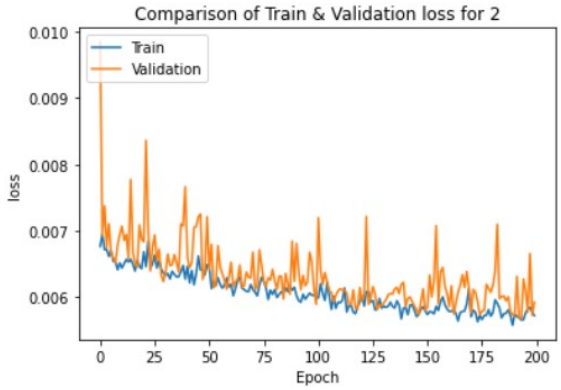

Figure 5.14: Training and validation loss for "mse" loss function

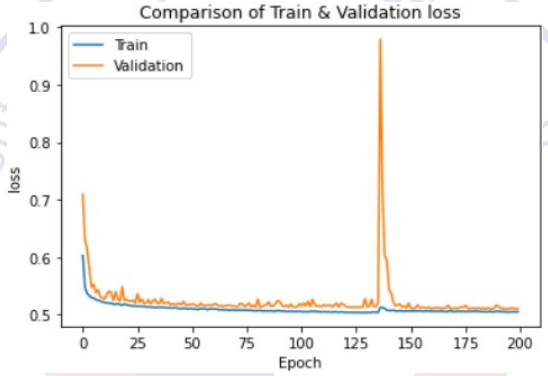

Figure 5.15: Training and validation loss for "Binary cross entropy" loss function

As it is clear from above figures, the loss function "Binary cross entropy" resulted in the smoother convergence of model. There were less ripples in the graph of loss versus epochs, so it was chosen as the loss function.

Table 5.3: Comparison of Autoencoder parameters before and after modification

Parameters	**SRGAN**	
	Original	**After Modification**
Total Parameters	6,620,419	6,730,819
Trainable Parameters	6,619,651	6,730,051
Non-Trainable Parameters	768	768

5.3 gives insight to variation in parameters of the model before and after including the feature extraction layer. It can be observed that no. of parameters got increased after the modification.

5.3 Performance Comparison

This section focusses on comparing the results of the original and modified sub models individually. The PSNR and SSIM metrics are used for comparing the results. Lastly, the PSNR and SSIM metrics are also included after the picture is passed through the proposed model.

5.3.1 Deblurring Network (SRGAN)

Table 5.4: Comparison of PSNR and SSIM after running SRGAN for 5 epochs and 10 epochs for both the Original SRGAN and Modified SRGAN

Metrics	**Original SRGAN**		**Modified SRGAN(10 Epochs)**
	After 5 Epochs	**After 10 Epochs**	
PSNR (image 1)	50.43	53.003	54.64
SSIM (image 1)	0.995	0.998	0.999
PSNR (image 2)	51.67	52.46	53.07
SSIM (image 2)	0.994	0.999	0.999
PSNR (image 3)	51.997	52.60	53.09
SSIM (image 3)	0.993	0.999	0.999
Average PSNR	51.99	52.69	53.60
Average SSIM	0.994	0.999	0.999

5.3.2 Saturation Handling Network (ESRGAN)

Table 5.5: Comparison of ESRGAN performance before and after modification

Metrics	**ESRGAN**	
	Original	**After Modification**
PSNR (image 1)	29.75	29.96
SSIM (image 1)	0.57	0.70
PSNR (image 2)	35.03	36.21
SSIM (image 2)	0.92	0.96
PSNR (image 3)	33.12	33.40
SSIM (image 3)	0.68	0.75
Average PSNR	32.64	33.19
Average SSIM	0.72	0.80

5.3.3 Denoising Network

Table 5.6: Comparison of PSNR and SSIM for the Denoising network

Metrics	Denoising Autoencoder	
	Without Feature Extraction Layer	**With Feature Extraction Layer**
PSNR (image 1)	32.78	33.55
SSIM (image 1)	0.57	0.59
PSNR (image 2)	31.75	30.42
SSIM (image 2)	0.89	0.87
PSNR (image 3)	30.93	31.12
SSIM (image 3)	0.86	0.90
Average PSNR	31.82	31.70
Average SSIM	0.77	0.79

5.3.4 Proposed model

Table 5.7: PSNR and SSIM values of each Individual Model and Proposed Model

Metrics	Individual Models			Proposed Model
	After SRGAN	**After ESRGAN**	**After DAE**	
PSNR (image 1)	54.64	34.36	35.36	35.36
SSIM (image 1)	0.99	0.87	0.88	0.88
PSNR (image 2)	53.07	36.34	34.97	34.97
SSIM (image 2)	0.99	0.93	0.91	0.91
PSNR (image 3)	53.09	33.64	35.34	35.34
SSIM (image 3)	0.99	0.86	0.91	0.91
Average PSNR	53.60	34.78	35.22	35.22
Average SSIM	0.99	0.89	0.90	0.90

5.4 Inferences drawn from the results obtained

In this section, the inferences drawn from the performance comparison is discussed in detail. As seen in 5.4, the PSNR values increase as the number of epochs increases, indicating that the picture quality increases. Even the SSIM values increases marginally. The SSIM values are in the order of 0.99, indicating near to perfect structural similarity.

The table also compares the PSNR and SSIM values before and after the modification of the SRGAN architecture. The modified SRGAN model has increased number of layers in both the generator and discriminator blocks. Here again the PSNR values increases which indicates that picture quality is much better. SSIM values still remain in the order of 0.99.

5.5 compares the PSNR and SSIM values before and after modification of the ESRGAN architecture. The modified model has 2 layers of ESRGAN and a fusion block

after each layer and a final fusion block after that, thus creating a total of 3 fusion blocks which has significantly improved the performance of the original ESRGAN model.

5.6 draws comparison between the PSNR and SSIM values of the Denoising Auto encoder with and without the feature extraction layer. It can be inferred from the values that Feature Extraction Layer improves the performance of the Denoising network even though the visual appeal appears similar.

5.7 shows the PSNR and SSIM values of the images after passing through each network, one after the other.

By comparing the visual quality of the output images along with the PSNR, SSIM values, it can be seen that the proposed model is performing better than the original, individual models being used. The blur, noise and saturation levels have significantly reduced as compared to the original input images passed through the proposed model.

Chapter 6

Conclusion and Future Scope

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In this work, we solved the problem of image deblurring by breaking down the problem into three sub problems i.e image deblurring, outlier handling and image denoising.We have proposed a three stage architecture which contains three unique parts.

All three models were trained independently to optimise their performance. For deblurring of images, a variant of Generative Adversarial Networks known as Super Resolution GAN is used. The deblurred image is then passed on to modified ESRGAN to handle the outliers present in the deblurred image.The image while passing through the deblurring and saturation handling network acquires noise which is denoised by the Convolutional Denoising Autoencoders.

6.2 Future Scope

Even though the purpose of this study has been satisfactorily achieved, there are certain limitations, too. A major drawback of SRGAN is that it fails to recover textures present in the image. For instance, small text appears distorted or smooth which is not desirable. Textures in the human face such as fine lines could cause a completely different human face to be generated. Considering that the first sub-model of the proposed model is the deblurring model, undesirable images could be fed to the subsequent sub-models which could lead to garbage images being created. The Saturation Handling Module is handling a good amount of saturation but ends up generating noise in the output image due to a double layer of ESR-GAN. While the fusion blocks reduce the generated noise to some degree, they do not completely eliminate it. Thus, tuning of the model to further reduce is necessary. Our denoising model was trained for removing noise which is recognisable by the human eye such as gaussian noise, speckle noise, impulsive noise etc. But there are many noise which is not recognizable from naked eye and is feeble and invisible. The latest adversarial attack security breaches on digital images is one such feeble and invisible noise. Therefore a training set which constitutes feeble and invisible noises is required to tackle such situations.

6.3 Learning Outcomes of the Project

- Gained in depth knowledge about the process of image deblurring
- Understood and implemented state of the art GAN models
- Understood and implemented state of the art saturation handling networks
- Understood and implemented state of the art Image denoising architectures
- Learnt tensorflow framework

BIBLIOGRAPHY

[1] C. Ledig, L. Theis, F. Huszár, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," pp. 105–114, 2017. DOI: 10.1109/CVPR.2017.19.

[2] X. Wang, K. Yu, S. Wu, *et al.*, "Esrgan: Enhanced super-resolution generative adversarial networks," in *The European Conference on Computer Vision Workshops (ECCVW)*, 2018.

[3] N. Divakar and R. Venkatesh Babu, "Image denoising via cnns: An adversarial approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[4] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: Blind motion deblurring using conditional adversarial networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8183–8192. DOI: 10.1109/CVPR.2018.00854.

[5] X. Xu, J. Pan, Y.-J. Zhang, and M.-H. Yang, "Motion blur kernel estimation via deep learning," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 194–205, 2018. DOI: 10.1109/TIP.2017.2753658.

[6] B. Zhao, W. Li, and W. Gong, "Deep pyramid generative adversarial network with local and nonlocal similarity features for natural motion image deblurring," *IEEE Access*, vol. 7, pp. 185 893–185 907, 2019. DOI: 10.1109/ACCESS.2019.2956947.

[7] S. Zheng, Z. Zhu, J. Cheng, Y. Guo, and Y. Zhao, "Edge heuristic gan for non-uniform blind deblurring," *IEEE Signal Processing Letters*, vol. 26, no. 10, pp. 1546–1550, 2019. DOI: 10.1109/LSP.2019.2939752.

[8] S. Lim, H. Park, S.-E. Lee, S. Chang, B. Sim, and J. C. Ye, "Cyclegan with a blur kernel for deconvolution microscopy: Optimal transport geometry," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 1127–1138, 2020. DOI: 10.1109/TCI.2020.3006735.

[9] B. Lu, J.-C. Chen, and R. Chellappa, "Uid-gan: Unsupervised image deblurring via disentangled representations," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 2, no. 1, pp. 26–39, 2020. DOI: 10.1109/TBIOM.2019.2959133.

[10] T. Kaneko and T. Harada, "Noise robust generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[11] ——, "Blur, noise, and compression robust generative adversarial networks," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13 574–13 584. DOI: 10.1109/CVPR46437.2021.01337.

[12] H. Tomosada, T. Kudo, T. Fujisawa, and M. Ikehara, "Gan-based image deblurring using dct loss with customized datasets," *IEEE Access*, vol. 9, pp. 135 224–135 233, 2021. DOI: 10.1109/ACCESS.2021.3116194.

[13] W. Niu, K. Zhang, W. Luo, and Y. Zhong, "Blind motion deblurring super-resolution: When dynamic spatio-temporal learning meets static image understanding," *IEEE Transactions on Image Processing*, vol. 30, pp. 7101–7111, 2021. DOI: 10.1109/TIP.2021.3101402.

[14] J. Dong and J. Pan, "Deep outlier handling for image deblurring," *IEEE Transactions on Image Processing*, vol. 30, pp. 1799–1811, 2021. DOI: 10.1109/TIP.2020.3048679.

[15] A. Karaali, N. Harte, and C. R. Jung, "Deep multi-scale feature learning for defocus blur estimation," *IEEE Transactions on Image Processing*, vol. 31, pp. 1097–1106, 2022. DOI: 10.1109/TIP.2021.3139243.

[16] M. Chang, C. Yang, H. Feng, Z. Xu, and Q. Li, "Beyond camera motion blur removing: How to handle outliers in deblurring," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 463–474, 2021. DOI: 10.1109/TCI.2021.3076886.

[17] X. Zhang, R. Wang, D. Chen, Y. Zhao, and W. Gao, "Handling outliers by robust m-estimation in blind image deblurring," *IEEE Transactions on Multimedia*, vol. 23, pp. 3215–3226, 2021. DOI: 10.1109/TMM.2020.3021989.

[18] Z. Zhao, B. Xiong, S. Gai, and L. Wang, "Improved deep multi-patch hierarchical network with nested module for dynamic scene deblurring," *IEEE Access*, vol. 8, pp. 62 116–62 126, 2020. DOI: 10.1109/ACCESS.2020.2984002.

[19] R. Bhatt, N. Naik, and V. K. Subramanian, "Ssim compliant modeling framework with denoising and deblurring applications," *IEEE Transactions on Image Processing*, vol. 30, pp. 2611–2626, 2021. DOI: 10.1109/TIP.2021.3053369.

[20] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[21] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, 2018.

[22] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. DOI: 10.1016/j.neunet.2014.09.003. [Online]. Available: https://doi.org/10.10162Fj.neunet.2014.09.003.

[23] D. Bank, N. Koenigstein, and R. Giryes, *Autoencoders*, 2020. DOI: 10.48550/ARXIV.2003.05991. [Online]. Available: https://arxiv.org/abs/2003.05991.