EAIDGAN: Edge and Attention-Based Image Deblurring by Generative Adversarial Network

Mifta Juneidi Umer

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September 2021

Adama, Ethiopia

EAIDGAN: Edge and Attention-Based Image Deblurring by Generative Adversarial Network

By: Mifta Juneidi Umer

Advisor: Dr.Worku Jifara

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September 2021

Adama, Ethiopia

Declaration

I hereby declare that this Master Thesis entitled " **Edge and Attention-Based Image Deblurring by Generative Adversarial Network**" is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation

Mifta Juneidi Umer	_____	_____
Name	Signature	Date

Recommendation

I, the advisor(s) of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Edge and Attention-Based Image Deblurring by Generative Adversarial Network**" prepared under my/our guidance by **Mifta Juneidi Umer** submitted in partial fulfillment of the requirements for the degree of Master's of Science in Computer Science and Engineering. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Dr. Worku Jifara (Ph.D.) _____ _____

Major Advisor Signature Date

Approval Page

I/we, the advisors of the thesis entitled "**Edge and Attention-Based Image Deblurring by Generative Adversarial Network**" and developed by **Mifta Juneidi Umer**, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Major Supervisor	Signature	Date
Co-Supervisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by ---------- (Name of student) have read and evaluated the thesis entitled "---------- ---------- ---------- ----------" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in ----------.

Chair Person	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Chair Person	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

ACKNOWLEDGMENT

Above all, I give my extraordinary thanks to the Almighty Allah for giving me health and strength through the thesis journey.

I want to express special gratitude to my advisor **Dr.-Worku Jifara** for his persistent guidance, support, and suggestion throughout the thesis work from starting up to the end. I want to express my thanks to Prof. Yun Koo Chung (Ph.D.) head of the Computer Vision and Robotics SIG for his persistent supervision and suggestion until the completion of the thesis.

I would like to thankAnteneh Tilaye (MSc.) for his support and advice by reading my thesis. I want to thank CVR SIG members for their support throughout this work.

Last but not least, My sincere thanks also go to all my family: My father Juneidi Umer, My mom Bedriya Abdullahi, My sister Hindiya Juneidi, My brother Gena Areba and close companions for their support and advice.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CODE SNIPPETS

LIST OF ACRONYMS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
Conv-nets	Convolutional Neural
CPU	Central Processing Unit
C ++	C-plus plus programming language
CV	Computer Vision
DCGAN	Deep Convolutional Generative Adversarial Network
DL	Deep Learning
GAN	Generative Adversarial Network
FPN	Feature Pyramid Network
GB	Gigabyte
GPU	Graphic Processing Unit
I2I	Image to Image
IDE	Integrated Development Environment
L1	Manhattan Distance or L1 Norm
L2	Euclidean Distance or L2 Norm
LSGAN	Least Square Generative Adversarial network
LWGAN	Wasserstein Generative Adversarial Network Loss
MSE	Mean Squared Error
MAE	Mean Absolute Error
MS	Microsoft
OS	Operating System
PGAN	Progressive Generative Adversarial Network
PSNR	Peak Signal to Noise Ratio
RAM	Random Access Memory

ReLU	Rectified Linear Unit
ResNet	Residual Blocks
RGB	Red Green Blue
RQ	Research Question
SAGAN	Self-Attention Generative Adversarial Network
SRGAN	Super-Resolution Generative Adversarial Network
SSIM	Structural Similarity Index Measure
TPU	Tensor Processing Unit
WGAN	Wasserstein Generative Adversarial Network
2D	2-Dimensional

LIST OF NOTATIONS AND SYMBOLS

Notation	Meaning
A	Indicate given image is domain A
B	Indicate given image is domain B
Dx	Discriminator X
Dy	Discriminator Y
X	Sample Image from category A
Y	Sample Image from category B
$X \rightarrow Y$ *or* $(Y \rightarrow X)$	Translation from domain X to domain Y or (Y to domain X)
Z	Noise data sample
θG	Generator Parameter
θD	Discriminator Parameter
et.al	and others
$\hat{S}(t)$	an image corrected by the gamma value

ABSTRACT

The image is a representation of something or thought we're imagining in our heads. Since one image tells a hundred words peoples uses an image to exchange information on different communication platform while, data scientist uses the image to investigate research on object detection and classification, action recognition, image generation, image enhancement and for other computer vision applications. To use an image for exchanging information and in computer vision applications the image must have good quality. However, image quality is frequently degraded by factors such as blur. Blur is a part of artifacts that occur during the image acquisition time. An image deblurring mechanism is required before using the blurry image for computer vision applications. Image deblurring is an essential and challenging low-level vision issue that aims to get a sharp image from the blurred image. In the past several works have been conducted on image deblurring by using the generative adversarial network on the paired dataset. However, those methods cannot deliver satisfactory outcomes when the blur size is large since they rely only on local convolution filters. In this work edge and attention-based image deblurring by the generative adversarial network is proposed to mitigate these issues. The architecture utilizes a feature pyramid network as the core building block of the generator since it can simply integrate with many pre-trained backbones. To improve the execution of the network with small computational complexity EfficientNet-B7 is utilized as a feature extractor in the generator network. Further, to handle a large size blur kernel without increasing parameters the channel and self-attention mechanism was utilized in this work to capture long-range dependencies between channels and pixels instead of relying only on local convolution filters. Moreover, Edge and least square loss are introduced to preserve the edge information and to stabilize the training respectively. Broad subjective and quantitative evaluations illustrate the striking victory of the proposed system against existing methods. The peak signal to noise ratio and structural similarity index measure reaches over 26.174 and 0.8406 respectively which outperforms the other comparative works. Average human assessment study has appeared that this research exceeds expectations at 25% compared to DeblurGAN-v2. This work concludes that adding EfficientNet-B7, self-attention, channel attention, edge, and least square loss does improve the execution of the deblurring network.

Keywords: *EfficientNet-B7, Generative Adversarial Network, self-attention, channel attention, Image deblurring*

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

The image could be a representation of something or thought we're imagining in our heads. Due to the growth of information technology, the importance of an image in people's daily life is expanding at a breakneck pace. People share the photo for each other by using different communication platforms to exchange information while, data scientist uses the image to investigate research on object detection and classification, action recognition, image generation, image enhancement and for other computer vision applications. To use an image for exchanging information and in computer vision applications the image must have good quality. However, image quality is frequently degraded by factors such as blur. Blur is a one-part artifact that occurred during the image acquisition time. It can be appeared by flaws during the image acquisition time such as camera shake, object motion, camera out-of-center, photos were taken with hand-held cameras.

A blurred image is an image in which the detailed information of an image is not fully visible due to its lack of a distinct shape (Nah et al., 2017). This loss of image details will result in a significant decrease in image quality and it will have a basic effect on the human visual experience, will pose a challenge to subsequent computer vision analytics, and will be incapable to meet the necessities for picture quality execution markers in industrial production (Kupyn et al., 2019). A blurry image E expressed can be expressed as:

$$E = K * S + N \qquad (1.1)$$

Where N, S, and K are representing the additional noise, the idle sharp picture, the blur kernel matrices respectively, and $*$ denotes convolution (Nah et al., 2017). The purpose of deblurring is to obtain the sharp picture S from the blurred picture E. Based on blur kernel blurry removal algorithm can be classified as blind and non-blind deblurring. A non-blind deblurring algorithm works with a blurry image that has a known blur size or kernel. Whereas, a blind deblurring mechanism is based on an unknown blur size or kernel assumption (Chakrabarti, 2016). Since real-world blurs found in an image is non-uniform and too complicated, how to obtain sharp images from blurred images is an issue.

With the current success of deep learning, numerous research works have employed deep learning architectures in image generation and image to image translation (I2I). Especially After the coming of the generative adversarial network the image generation and image-to-image translation task become easy. I2I is a branch of domain transfer that aims to map one class to another by using generative adversarial network architecture. Such domain transfer could be used in different areas including image colorization, image inpainting, style transfer, and image deblurring. Image deblurring is a sub-problem of an image-to-image translation that aims to obtain a sharp image from the blurry image.

To fully comprehend the Generative adversarial network, we must first provide a brief overview of the following topics.

1.1.1. Generative Adversarial Network (GAN)

The word generative refers to the fact that how these networks can learn data distribution to synthesize fake data that are similar to real image (Goodfellow et al., 2020). A GAN is a family of deep learning in which contains two neural networks known as generators and discriminators compete in a min-max game framework (Goodfellow et al., 2020; Radford et al., 2016). The main aim of the generator network is to create new data samples from random noise close to that of the training set, while discriminator networks try to classify actual samples from fake ones as appeared in figure 1.1. The Generative network converges when the generator and the discriminator network reach the Nash equilibrium. At Nash equilibrium discriminator randomly classifies real and synthesized samples with an accuracy of 50.

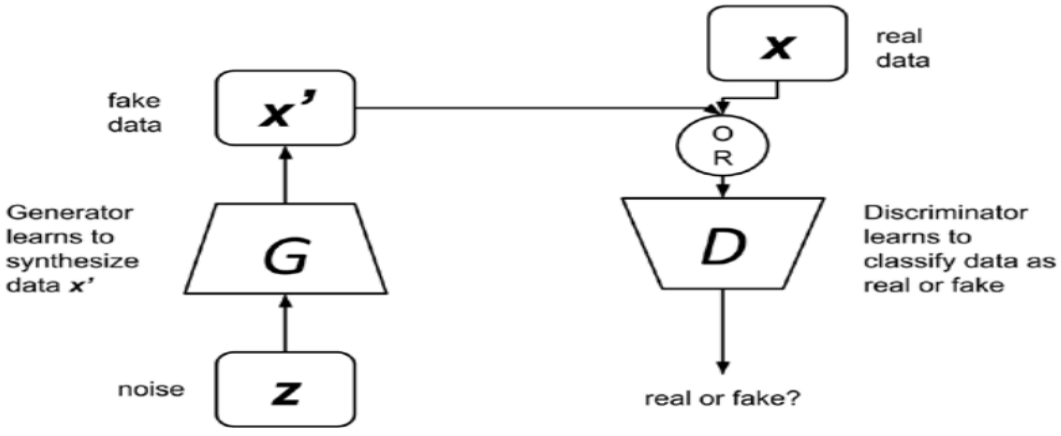


Figure 1. 1 Architecture of GAN

Source: Adapted from (Goodfellow et al., 2020)

Conditional GAN (CGAN): The CGAN (Mirza & Osindero, 2014) is an expansion of the GAN (Goodfellow et al., 2020) where the discriminator and generator are conditioned on a few extra data **Y. Y** could be a condition variable such as information from other domains or class labels that would allow generating a particular data distribution specified by the conditioning variable. CGAN is described in detail with architecture in section 2.2.3.

Recent works (Kupyn et al., 2019; Pan et al., 2016) use the CGAN to restore sharp images from blurred images. However, this work depends on the nearby features and overlooks long-range dependencies from the non-local features and this limits the capability of the network to learn contextual information from clean areas to restores sharp photos from the blurry photo when the blur size is large. This work tries to solve these problems by extending the solutions presented in these works.

1.2. Motivation of the Study

Recently deep learning practically delivered stunning achievements due to the abundance of the enormous amount of data available nowadays. The quality of data is the fundamental thing in deep learning. The quality of data has a critical impact on the execution of deep learning: high-quality data can lead to high execution while poor data can reduce the execution of deep learning. Data (image data) can be degraded by factors such as blur which occurs by camera shake, object motion, and camera out-of-focus. These issues can be tackled by an image deblurring mechanism. After the first paper on GAN by Goodfellow (Goodfellow et al., 2020), GAN has been made critical advancements and performance in many applications such as image-to-image translation, realistic image generation, and video-to-video translation. This is applicable in photo editing, game design, and multimedia content generation.

By adopting the image-to-image translation idea, many researchers propose many approaches for image deblurring networks to obtain sharp images from blurred images. However, previous works rely on the local convolution features and do not care about long-range dependencies from the non-local features and this makes it difficult to obtain an idle sharp image from a blurry image when the blur size in the image is large. These issues are still open for further investigation in the image deblurring problem.

1.3. Statement of the Problem

Computer vision is one of the Artificial intelligence subfields that take images or video as input and works through image, recognition segmentation, object detection, and image super-resolution. To do all those actions the clarity of the image is very important. However, the quality of an image can be degraded by factors such as blur. Since a blurry image does not have a clear edge it is difficult to identify the objects in the image when the blur size in the image is large. Due to some information and details are lost in a blurry image, it is difficult to use the blurry image for computer vision applications such as object detection, image segmentation, classification, generation, and recognition. Therefore, an image deblurring mechanism is required before using the blurry image for computer vision applications. Recent works (Kupyn et al., 2019; Nah et al., 2017) proposes image deblurring architecture based on CGAN. However, these works rely on the local convolution features and do not care about long-range dependencies from the non-local features and this makes it difficult to obtain an idle sharp image from a blurry image when the blur size in the image is large.

To overcome this challenge edge and attention-based image deblurring by the GAN method is proposed in this paper. Architecture is designed based on attention mechanism (self and channel) and edge loss to make the network aware of global information between channel and pixel which helps the network to handle large size blur kernel.

1.4. Research Questions

The goal of this work was to find an answer to the following question.

RQ1: What is the effect of adding an attention mechanism regarding

improving deblurring quality?

RQ2: What are the appropriate learning constraints that improve deblurring quality?

RQ3: What is the impact of image deblurring in computer vision applications?

1.5. Objectives

1.5.1. General Objective

The general objective of the study is to develop an edge and attention-based image deblurring by GAN.

1.5.2. Specific Objectives

The following are specific objectives under the proposed research to achieve the main objective.

- To review concepts and related works on image deblurring.
- To add attention mechanism (channel and self-attention) to generative adversarial network.
- To integrate EfficientNet-B7 as a feature extractor to the generator in GAN.
- To identify the learning parameter of the designed model that improves the quality of an image.
- To train the proposed model on the available dataset.
- To assess the model performance using standard evaluation metrics on test data.

1.6. Significance of the Research

Before using an image for a different purpose the sharpness of an image is important. This study should be applied to different scenarios and serve as a primary-steps for other applications such as object detection, image recognition, image classification, photo editing, face recognition, and video surveillance. Generally, image deblurring architecture presented in this study has been applied to different applications listed above with success in restoring a sharp image from a blurred image. Future works might benefit from adapting and customizing this work on a large dataset and utilizing it for numerous applications.

1.7. Scope and Limitation of the Study

1.7.1. Scope of the Study

The objective of this research is to create the architecture for image deblurring. The implementation of the edge and attention-based image deblurring by GAN is to improve the image deblurring tasks. Accessing the quality of the deblurred image results was also intentionally done. The proposed work conditioning is the generator by feeding the blurred image as input rather than random noise. As a result, the scope starts from developing an edge and attention-based image-deblurring network to evaluate the results in contrast with other previously done approaches.

1.7.2. Limitation of the Study

This work does not cover the following due to tight schedules and resources limitations.

- The style transfer is not part of this work. The proposed model is trained to translate blurred images into sharp images.
- Image super-resolution is not part of this work

1.8. Application of Results

Image deblurring is a pre-request to improve the sharpness of the image before it is processed in many application areas such as computer vision, computed tomography, classification, detection, remote sensing, video surveillance, and face recognition. Generally, image deblurring is applicable in the different areas where an image is used.

1.9. Organization of the Thesis

This thesis has been organized into seven-chapter as follows:

Chapter Two: discusses the background literature related works regarding image-to-image translation and the theoretical framework of Deep Learning and Generative Adversarial Network.

Chapter Three: presents the research methodology, methods, and techniques used to develop the solution, data collection mechanism, preprocessing, and annotation techniques, design tools, and model and evaluation methods are discussed.

Chapter Four: describes the proposed model in detail. The proposed Architecture, loss functions, and pseudocode for training have been made.

Chapter Five: Explains how the proposed solution is implemented and the working environment setup, edge and attention-based image deblurring by GAN implementation, with training implementation, described using snip code.

Chapter Six: explains the obtained results from the edge and attention-based image deblurring by GAN and compares the result with other work.

Chapter Seven: summarizes and concludes the work along with the result and it explores the open issues in the research.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORKS

2.1. Introduction

Computer vision could be a scientific field that centers on how computers make an understanding of video and advanced pictures. It is a department that deals with the classification of objects, as well as to object recognition, detection, and segmentation. It tries to automate computers to process and identify items in videos and images like humans. Because of the huge volume of data available, today computer vision applications have evolved from simple data processing to machine learning in recent years.

Machine learning is the huge research area that makes computers learn and grow through experience from data. Depending on different learning capabilities machine learning techniques can be categorized into reinforcement, unsupervised, and supervised learning. Supervised learning is a basic machine learning algorithm that works on labeled data. Unsupervised learning is another machine learning technique that can work with unlabeled data. Unsupervised learning's primary goal is to group data points that are similar in a dataset without the label. Reinforcement learning is a family of machine learning that an agent gains knowledge from its surroundings by performing an action and observing the results. Machine learning plays a great role in the advancement of computer vision applications but it is difficult to process complex data such as image and video by machine learning algorithms and to work with such complex data a branch of machine learning techniques called deep learning was used.

Deep learning is a family of machine learning that makes computers learn the complex representations needed for detection or classification from the row data (Lecun et al., 2015). The term deep indicates the number of layers in the network through which the input data is transformed and helps deep learning solve more complicated tasks. Deep learning algorithms pass the input data through layers and each layer is extracts features and passes them to the layer after them (Lecun et al., 2015). Artificial neural network (ANN) is the deep learning algorithm that gives the association between input and output data by artificially mimicking the physiological structure and functioning of human brain structures (Kwon, 2011). Convolutional Neural Network (CNN) was introduced in 1989 by Y. Le Cun et.al to recognize handwritten numbers (LeCun et

al., 1999). CNN has had amazing results over the past time in different fields related to image processing, pattern recognition, and video processing. CNN is effective than ANN since they have smaller parameters than an artificial neural network. This helps researchers to use larger models to solve complex problems, which was impossible in classic ANN. The CNN has seen dramatic development after Alex-net (Gonzalez, 2007) win the ImageNet contest in 2012 using the CNN image classifier network. Today many CNN effective models such as GAN are designed for different tasks. Generative adversarial network are new emerging deep learning techniques where the models are trained to learn real data distribution to synthesize fake data that are similar to real data. Let's see the detail of the GAN in the next section.

2.2. Generative Adversarial Network (GAN)

The GAN introduced by Goodfellow in 2014 (Goodfellow et al., 2020) model contains two independent models called discriminator model D and generator model G that are trained simultaneously as depicted in figure 1.1. The objective of a generator is to capture actual data and generate fake samples that look like real data. Discriminator plans to recognize fake samples from the actual input data. This helps the generator to understand the real data pattern and generate samples closer to real data. G and D are functions represented by a multilayer network with their independent parameters called θG and θD respectively (S. Wang, 2017). Discriminator network trained to discriminate the real training samples and fake generated samples. The generator improves the quality of fake samples by using criticism from the discriminator to trick the discriminator network. This situation is continued until both networks reach Nash equilibrium.

2.2.1. Principles of GANs

GAN is practically equivalent to a counterfeiter-police situation, where the generator act as a counterfeiter and the discriminator act as police (Rodney & López, 2019). The police are instructed on how to identify whether a currency is either original or fake at the academy. Tests of real currency from the bank and fake currency from the counterfeiter are utilized to prepare the police. In any case, from time to time, the counterfeiter will try to imagine that he printed genuine currency. At first, the police will not be tricked and will tell the counterfeiter why the currency is fake. Taking into thought this criticism, the counterfeiter sharpens his aptitudes once more and tries to create new fake currency. As anticipated, the police will be able to both spot the money as fake and legitimize why the dollar bills are fake. This handle proceeds uncertainly, but it'll come

to a point where the counterfeiter has aced the creation of fake currency to the degree that the fakes are undefined from genuine currency – even to the foremost exceedingly practiced of police. The counterfeiter can at that point interminably print fake currency without getting caught by the police as they are not identifiable as fake currency.

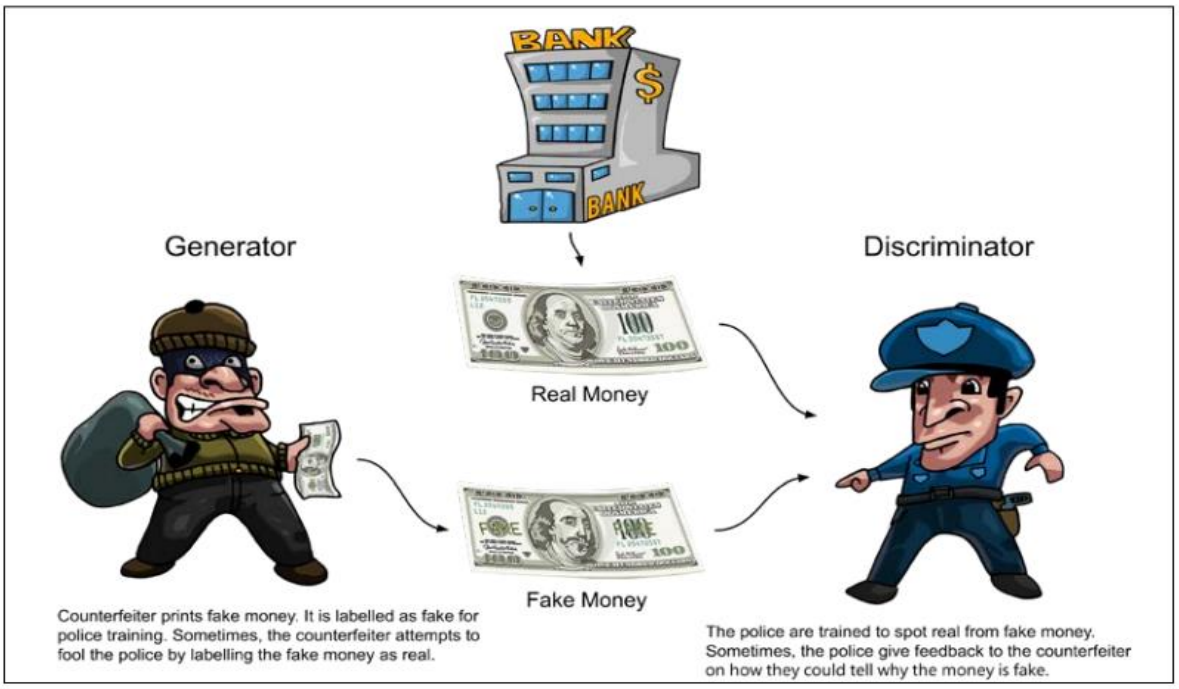


Figure 2. 1 GAN discriminator and generator are similar to counterfeiters and police.

Source: Adapted from (Rodney & López, 2019)

2.2.2. GAN Training

In deep learning training, a model means determining better weight values from examples. The training process is searching for a good weight that is enough at solving the given problem. The same is true for GAN, training is all about the process of learning real data distribution to generate the fake ones. Training of GANs is searching the parameters of the generator which confuse the discriminator and finding discriminator parameters that maximize classification accuracy (Creswell et al., 2018). During training discriminator parameters are fixed while parameters of a generator are updated (Goodfellow et al., 2020). The discriminator is trained for some number of iterations and then the generator model is updated simultaneously with the discriminator model.

Let's see pseudocode for training GANs

For each training iteration *do*

A. Train Discriminator

a. Take an arbitrary batch from real samples: x
b. Take arbitrary noise vector z and generate a group of fake samples: G(z)=x*
c. Compute the losses for D(x) and D(x*) and backpropagate the total error to update parameters of Discriminator to minimize discriminator loss

B. Train Generator

a. Take arbitrary noise vector z and generate a group of fake samples: G(z)=x*
b. Compute classification loss for D(X*) and backpropagate the total error to update parameters of a generator to maximize the discriminator loss

End

GAN is based on a min-max game between discriminator and generator networks. Discriminator wins generator loses and vice versa. When the discrimination network and the generation network reach the Nash equilibrium, the generation network converges i.e., the network reaches the Nash equilibrium. At Nash equilibrium discriminator randomly classifies actual and fake data with an accuracy of 50%. In later times numerous many researchers have developed numerous GAN architectures to solve different computer vision problems. Let's describe some of the GAN architectures in the next section.

2.2.3. Generative Adversarial Networks Architectures

A. Deep Convolutional GAN (DCGAN)

DCGAN merges a GAN with a CNN as depicted in figure 2.2. It has been developed to stabilize GAN to train and synthesize high-quality images (Radford et al., 2016). DCGANs are based on fractionally stride convolution for generator and stride convolution for the discriminator to downsample and spatial up sample the resolution of an image instead of the pooling layer. To normalize the input and tackle the issue of vanishing gradient batch normalization (BN) is introduced in DCGAN. DCGANs uses ReLU and LeakyReLU as activation function to improve the network performance and to make the model learn quickly.

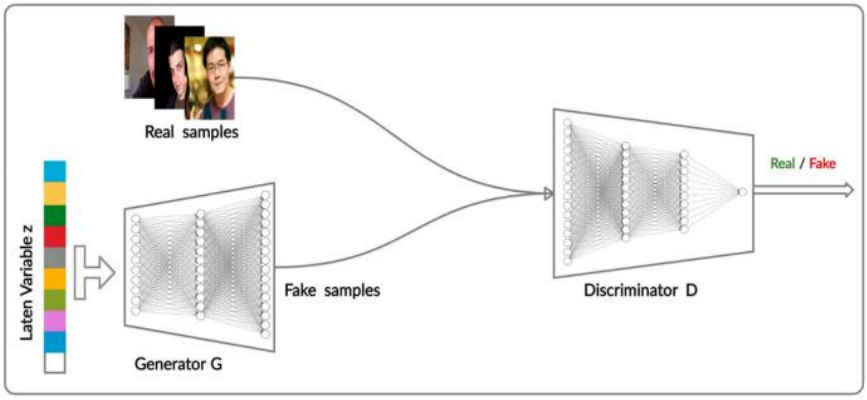


Figure 2. 2 Architecture of DCGAN

Source: Adapted from(Review & Alotaibi, 2020)

B. Conditional Gan (CGAN)

GAN was a mechanism to train generative models. GAN models can produce new plausible fake samples from real samples. However, there is no mechanism to control the sort of data being generated. The CGAN (Mirza & Osindero, 2014) was proposed by Mirza et al. to tackle the issue of conditional image generation. It conditions the generator by a class label such as data from other domains which allows generating a particular data image from the data distribution. The class label fades into the discriminator and generator network as appeared in figure 2.3 to make the generator synthesize only the target class of image given from the class label. This work enables researchers to do their work on interesting research areas such as I2I translation, style transfer, and image deblurring.

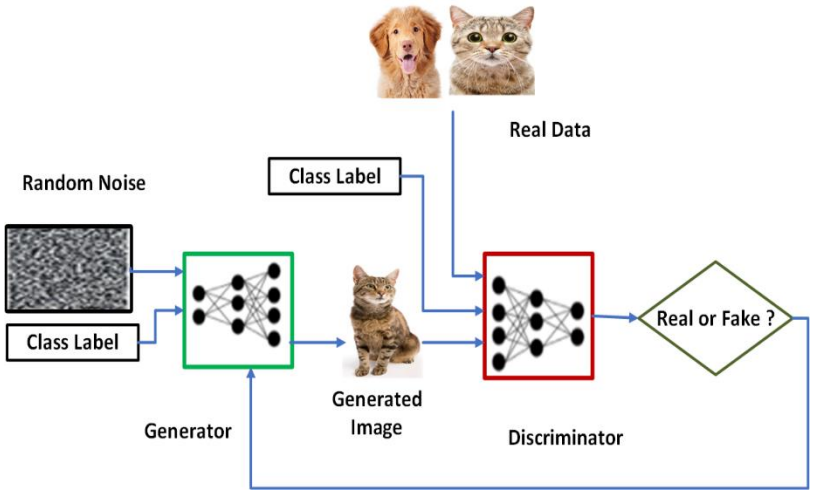


Figure 2. 3 Architecture of CGAN

D. Self-Attention GAN (SAGAN)

Recently, attention mechanisms are popular and most useful algorithms in deep learning. They are motivated by how humans pay attention to different parts of an image. In computer vision, the attention mechanism makes the network focus an important part of the image during image generation and this makes the network synthesize high-quality images. In 2019 Han Zhang et.al proposes SAGAN (H. Zhang et al., 2019) to make the quality of synthesized images better by incorporating a self-attention behavior into the GAN network. Due to the receptive field of the convolution filter is operating at a small region of the image it is difficult to model long-range dependency between image pixels by convolution filters only. To make long-range dependencies along image regions, SAGAN adds a self-attention behavior to the generator network (H. Zhang et al., 2019). SAGAN uses spectral normalization to minimize the number of computations and to improve performance. As depicted in Figure 2.4, input feature-map $X \in R^{C\times H\times W}$ is split into three new feature maps f(x), g(x) $\in R^{C\hat{}\times H\times W}$ ($C\hat{} = C/8$) and h(x) $\in R^{C\times H\times W}$. Next, f(x) and g(x) reshaped into $R^{C\hat{}\times N}$ ($N = H \times W$), carry out matrix multiplication between their transposed versions, and activated by SoftMax layer to calculate attention map. If A represents an attention map, then the attention map can be obtained as follows:

$$Ai,j = \frac{\exp(\mathrm{Sij})}{\sum_{i=0}^{n}\exp(\mathrm{Sij})} \quad , \text{where } \mathrm{Sij} = \mathrm{f(x)T.g(x)} \qquad (2.1)$$

Where Ai, j measures the impact of the i^{th} position on the j^{th} position. The likeness of the feature of any two positions contributes to the higher correlation between them. The final output can be obtained by performing matrix multiplication between the transposed version of the attention map (Aij) and h(x). The final output can be calculated by matrix multiplication between h(x) and the transpose of A (attention map) and added to input feature-map X as shown in equation 2.2.

$$K = Xi + \sum_{i=0}^{n} Aij.h(xi) \qquad (2.2)$$

Where Xi, Aij, and h(xi) are representing a set of input feature-map, attention map, and value respectively.

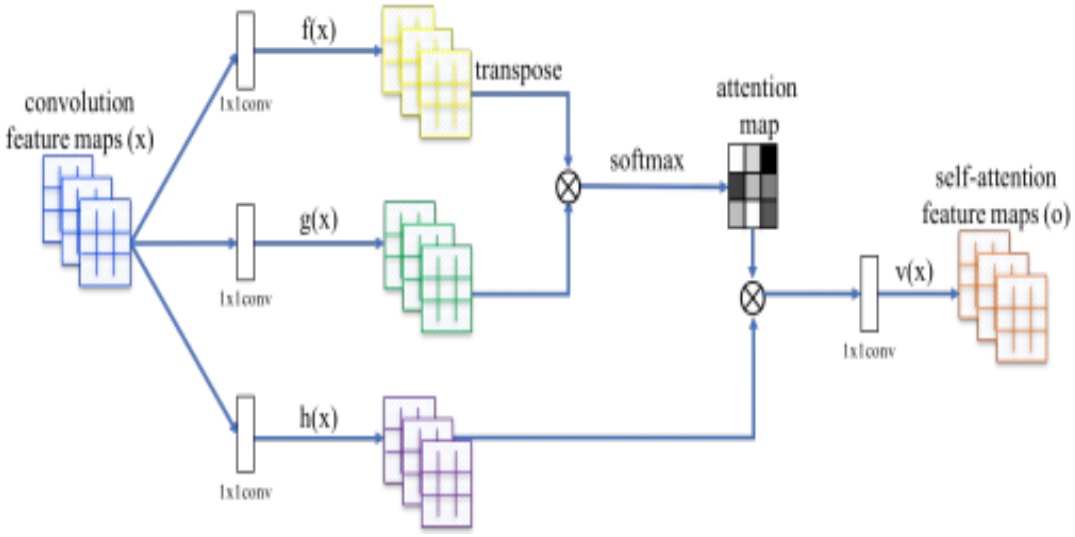


Figure 2. 4 Self-Attention GAN module. The ⊗ indicates matrix multiplication

Source: Adapted from(H. Zhang et al., 2019)

2.2.4. GAN Objective Functions

The objective functions used in GAN measure the gap between the actual data and fake data distribution. GANs have problems such as model collapse and vanishing gradient that will occur during training due to objective functions. These issues are solved by reformulating the objective (loss) function. To tackle the limitation of the initial GAN and to improve image quality many objective functions have been developed. Let's see some of them.

Adversarial Loss: Original GAN contains two networks that compete with each other by the cross-entropy loss function. The two-network loss is based on min-max loss i.e., win for one network is lose for another or vice versa. The discriminator is a function that takes input (x) and outputs single scalar D(y). D(y) implies the likelihood of input (x) is real or fake data. Similarly, generator (G) is a function whose input is random noise (z) is synthesized fake image looks real image. Discriminator and generator functions have their loss function as depicted on eq 2.3 and eq 2.4 respectively. Generator and discriminator can update their parameter until they reach to reach Nash equilibrium. Discriminator strives to maximize the likelihood of classifying real data as real and fake data as fake. Whereas, the generator strives to minimize the loss to trick the discriminator (Mathew et al., 2021).

$$LGAN(D) = Ex \sim \text{pdata(x)}\ [\text{logD(x)} + \text{Ez} \sim \text{P(z)}[\log\left(1 - \text{D}\left(\text{G(z)}\right)\right)] \qquad (2.3)$$

$$LGAN(G) = \mathrm{Ez} \sim \mathrm{P(z)}[\log\left(1 - \mathrm{D}\left(\mathrm{G(z)}\right)\right)] \quad (2.4)$$

Where, (Pdata, Pg) shows actual data distribution and generated data distribution (Review & Alotaibi, 2020).

Wasserstein GAN: Arjovsky (Arjovsky et al., 2017) et.al proposes WGAN to overcome the GAN mode collapse and model instability. WGAN calculates the similarity between generated and real data patterns using the Wasserstein distance or Earth Mover's (ME) distance rather than the Jensen Shannan divergence (Radford et al., 2016). The Earth Mover's (ME) distance is used to calculate the gap between actual data distribution and generated data distribution. Both discriminator and generator have their own cost as shown on eq 2.5 and 2.6 respectively.

$$LWGAN(D) = Ex \sim \mathrm{pdata(x)}[\mathrm{D(x)} + \mathrm{Ez} \sim \mathrm{P(z)}\ [\left(1 - \mathrm{D}\left(\mathrm{G(z)}\right)\right)] \quad (2.5)$$

$$LWGAN(G) = \mathrm{Ez} \sim \mathrm{P(z)}[\ \left(1 - \mathrm{D}\left(\mathrm{G(z)}\right)\right)] \quad (2.6)$$

Least Squares GAN: Mau et.al (S. Zhang et al., 2020)propose LSGAN in 2016 to tackle the vanishing gradient issue by adopting least square loss rather than cross-entropy. The authors stated that the cross-entropy loss function leads to vanishing gradient during updating generator using generated samples that are correct on decision boundary but distant from the actual data. As depicted in figure 2.5 (b) When fake samples (magenta) are used to modify the generator by convincing discriminators that they are from the actual data distribution, there will be no error Since they are on the right way of the boundary. However, these samples are far from the decision boundary and they need a mechanism to pull them close to the actual data (Mao et al., 2017). As a result, the authors propose LSGAN, which employs the least-squares (LS) objective function as the generator and discriminator loss function. The LS loss function pulls fake samples to the decision boundary, by penalizing samples that are on the right side decision boundary but, far from the decision boundary as depicted in figure 2.5 (b). It penalizes the generated images (in magenta), and it encourages the generator to synthesize samples near to decision boundary.

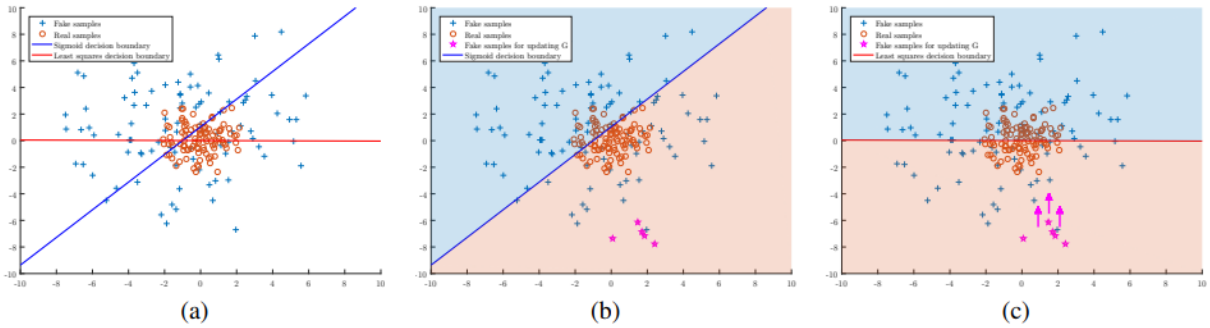


Figure 2. 5 (a): Choice boundaries of two-loss functions. (b): Choice boundaries for the cross-entropy loss function (c): Loss function based on least squares.

Source: Adapted from (Mao et al., 2017)

If b and a are labels for real and fake data, respectively, then the least square loss of generator and discriminator can be obtained as follows:

$$\min \mathrm{D}\ VLSGAN(D) = \frac{1}{2}E(x) \sim \mathrm{Pdata(x)}[(\mathrm{D(x)} - \mathrm{b})^{2}] +$$

$$\frac{1}{2}E(z) \sim \mathrm{Pz(z)}[(\mathrm{D(G(z))} - \mathrm{a})^{2}] \qquad (2.7)$$

$$\min \mathrm{G}\ VLSGAN(G) = \frac{1}{2}E(z) \sim \mathrm{Pz(z)}[(\mathrm{D(G(z))} - \mathrm{c})^{2}] \qquad (2.8)$$

Where c is the value that the generator uses to trick the discriminator, VLSGAN (D) is the discriminator objective function and VLSGAN (G) is the generator objective function.

Relativistic Discriminator: Montreal et.al introduces a relativistic loss to the discriminator (Jolicoeur-Martineau, 2019) in 2018 to make the quality of an image better. Relativistic loss calculates the probability of an actual image is more natural than a fake image and vice versa. The author states discriminator and generator cost functions as shown in Equations 2.9 and 2.10 respectively.

$$LD = -\mathrm{E(xr, xf)}\sim\mathrm{(P, Q)}[\log(\mathrm{sigmoid}\left(\mathrm{C}\left(\mathrm{(xreal)} - \mathrm{(xfake)}\right)\right)] \qquad (2.9)$$

$$LG = -\mathrm{E(xr, xf)}\sim\mathrm{(P, Q)}[\log(\mathrm{sigmoid}\left(\mathrm{C}\left(\mathrm{(xfake)} - \mathrm{(xreal)}\right)\right)] \qquad (2.10)$$

Where (xreal), xfake are the real and fake data samples respectively.

2.3. Image to image translation (I2I)

Suppose you take a selfie and wish to create it to look more artistic like a cartoonist's drawing how can a computer do this automatically? The I2I translation problem is the name given to such type of work. The I2I translation is a subproblem of computer vision in which tries to learn the transformation between two domains using a training dataset (Pang et al., 2021). The problem of I2I translation can be described as translating an image from a source domain, X, target domain, Y, for example, grayscale image to color image, color photo to edge-map. I2I translation algorithms capture special features of one image domain and figuring out how these features could be transformed into the other domain. There are two types of I2I translation methods: supervised and unsupervised depending on training data. Supervised I2I translation uses pairs of the image to translate source images to target images (Pang et al., 2021). Supervised I2I translation can translate synthesizing color photos from edge-maps, black and white to color images, and generating images from label maps as depicted in figure 2.6. Unsupervised I2I translation performs translation between source and target domain without paired input-output. I2I translation gets attention in the current time because it can be used for many computer vision applications such as style transfer, photo colorization, image deblurring, and image super-resolution.

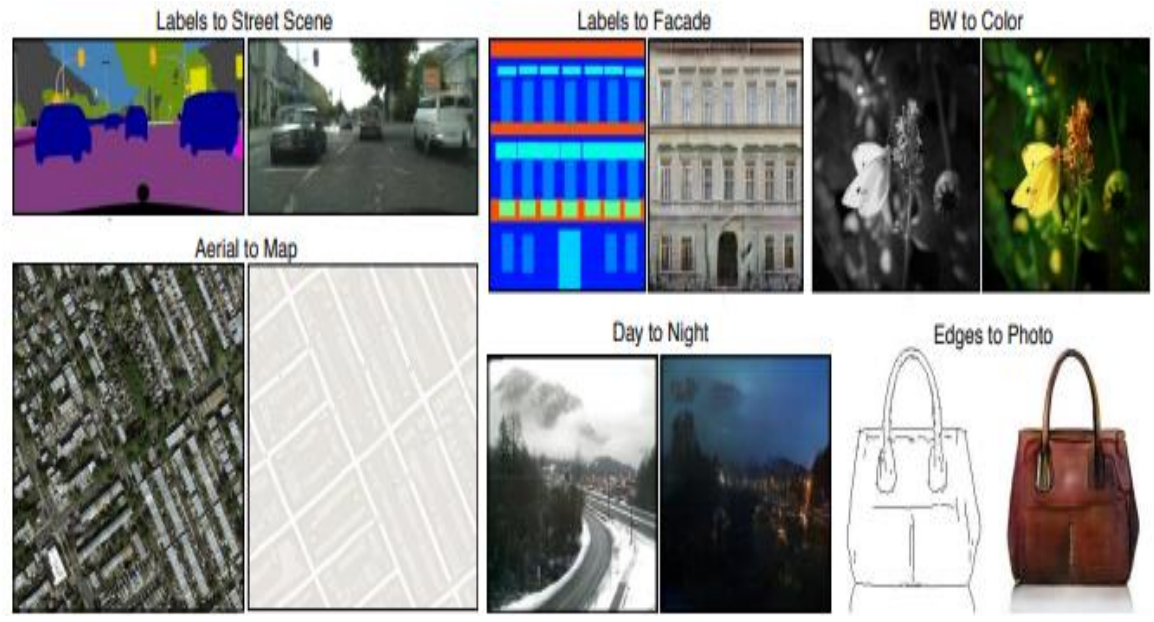


Figure 2. 6 examples of image-to-image translation applications

Source: Adapted from (Isola et al., 2017)

2.3.1. Style Transfer

Style transfer is the photo stylization problem where the network paints the content of one image with the style of another image. To alter the style of an image from one style to another the network takes two images content image as source and style image as a reference image by combining source and reference image the output looks like a source image but painted by the style of referenced image (Z. Xu et al., 2018) as shown on figure 2.7.

Figure 2. 7 original images, style image, and the synthesized image

Source: Adapted from (Z. Xu et al., 2018)

2.3.2. Image Super-Resolution

Image Super-Resolution is the low-level vision issue that seeks to produce photorealistic images (HR) images from low-resolution (LR) images (Ledig et al., 2017). Based on training data, image super-resolution learns a mapping between low-resolution image and high-resolution image information. Christian Ledig et.al (Ledig et al., 2017) propose SRGAN, which generates a high-

resresolution image with 4x up-scaling while preserving content information as depicted in figure 2.8.

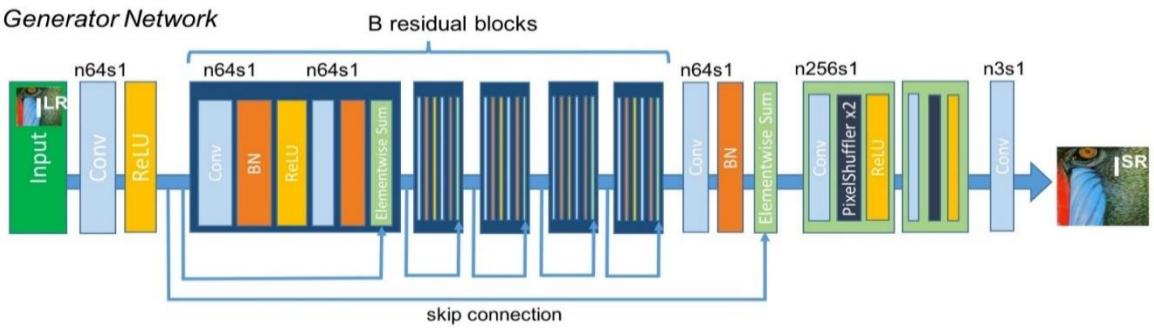


Figure 2. 8 Architecture of SRGAN

Source: Adapted from (Ledig et al., 2017)

2.2.3. Image Deblurring (ID)

Image deblurring is a difficult and fundamental problem that seeks to recover sharp images from blurred images (Kupyn et al., 2019; Nah et al., 2017). The goal of image deblurring is to learn a mapping function F between the blurry input image and the sharp image as depicted in figure 2.9. Many researchers have proposed numerous methods to address deblurring problems over the years, all of which use from classic to well-known deblurring schemes. The deblurring issue can be categorized into two classes based on the blur kernel **K**: blind deblurring and non-blind deblurring.

A. *Non-Blind Deblurring*

A non-blind deblurring algorithm works with a blurry image that has already a known blur size (kernel) (Koh et al., 2021). In this deblurring mechanism, the neural network can be used as a deconvolution network to get actual images from the blurry image. To obtain the actual image from a blurry image those algorithms use a known kernel at the deconvolution network. Since the blur size is already known, a suitable blurry image can be synthesized by convolving an actual image with a blur kernel (Koh et al., 2021). Hence, those algorithms are commonly trained on synthesized datasets. In general, non-blind deblurring algorithms cannot produce a good outcome when the size of the blur is large, and it is difficult to tackle the deblurring problem because the kernel of real blur is unknown.

B. *Blind Deblurring*

Blind deblurring is the inverse of non-blind deblurring; it is based on an unknown blur kernel assumption. Blind image deblurring is another image-deblurring problem that strives to obtain a

sharp image from a given blurred image (Kupyn et al., 2019; Nah et al., 2017) without knowing the blur size (kernel). Those methods learn the mapping function between blurry and real images. Non-blind deblurring algorithms are restricted to spatially unchanged blur, as different kernels required to identify a spatially variant blur are available. Hence, spatially changed blurs are mostly classified as blind deblurring problems.

Figure 2. 9 Sharp and blurry image pairs

2.4. Related Works

Chakrabarti et.al (Chakrabarti, 2016) have proposed a CNN algorithm that estimates blur kernel at patch level which is utilized in optimization framework for restoring the sharp image. The estimated kernel at the patch level to obtain the global kernel. After that, the global kernel is utilized as a deconvolution filter to obtain a sharp image from the blurry image. However, this work does not perform well when the size of the blur is large since it focuses on the patch level.

Nah et.al (Nah et al., 2017) offer the advantage of enabling generalized dynamic image deblurring at low latency, by iterative optimization stage involving the fitting of motion models designed by hand. But there are two limitations to their work. The first one is filters of CNN are limited to the receptive field and offer limited accuracy. The second one is the attempt to increase receptive field can increase the model parameter. This makes networks with large model sizes and a high computation cost and makes the model unsuitable to utilize in real-time applications.

Jinshan et.al proposes a deep learning framework for image deblurring by using dark channels prior (Pan et al., 2016). The authors used dark channel prior (the smallest RGB pixels value is likely to be zero) to deblur scenarios such as text, low-illumination images, and face. This work performs well when the dark channel between real and blurred images is high. However, their method does not perform well when bright pixels dominant in the image.

Xiangyu et.al proposes learning to enhance the blurry text and face by using GAN (X. Xu et al., 2017). To obtain a sharp image from a blurry image feature matching loss is added to CGAN. This work outperforms other prior works on text and faces images, however, it excludes other complex blurry objects.

DeblurGAN-v2 (Kupyn et al., 2019) was proposed by Kupy et.al by utilizing a relativistic average discriminator as a loss function to train the network. To make the network more flexible and compatible with a variety of backbones, as well as to balance performance and efficiency Feature Pyramid Network (FPN) is utilized as a generator building block. This work achieves competitive execution on a variety of efficiency benchmark datasets. However, the network relies on the local features (convolution kernels) this limits the network from learning long-range dependencies between pixels and channels which helps the network to synthesize sharp images when the blur size in the image is large.

Chunxue Wu et.al proposes deblurring architecture by using CycleGAN (Wu et al., 2020) to synthesize sharp images from blurred images. CycleGAN and PatchGAN discriminator was utilized in this work to obtain the sharp images from the blurry images. To measure the gap between real and generated images, perceptual and Wasserstein gradient penalty objective functions were used. However, this doesn't produce a promising result due to the loss of valuable information and an unstable train of a pair of networks.

2.5. Summary of Related Works

Table 2.1 illustrates a summary of related works.

Table 2. 1 Summary of related works

Related papers	**Dataset**	**Architecture**	**Evaluation metrics**	**Limitation**
Pan et al (Pan et al., 2016)	GoPro	CNN and dark channel loss	PSNR, SSIM	Does not perform well when bright pixels are dominant in the image
Ayan Chakrabarti (Chakrabarti, 2016)	Sun dataset	CNN Fourier transform (DFT)	Error ratio	Does not perform well the size of blur is large since the network is based on kernel estimation technique at the patch level.
Seungjun Nah et al(Nah et al., 2017)	GoPro	Multi-scale CNN	PSNR, SSIM	Complex architecture and hard to train. The attempt to handle large blur in an image is done only by maximizing the Convolution filter.
Yujin Zhang et al(X. Xu et al., 2017)	Text and face dataset	Conditional GAN	PSNR, SSIM	The network focus only on text and face image and excludes other complex blurry objects

Chunxue Wu et.al(Wu et al., 2020)	GoPro	Cycle-GAN	PSNR, SSIM	Doesn't produce a promising result, due to the loss of valuable information and an unstable train of a pair of networks
DeblurGAN-v2 (Kupyn et al., 2019)	GoPro	FPN with pre-trained InceptionResnet-v2 backbone	PSNR, SSIM	The network relies on the local convolution filters and ignores long-range relationships from the non-local features.

As appeared in the above table, researchers propose different architectures in previous work to obtain a sharp image from a blurred image. However, those works rely on local convolution filters to restore an image, hence using only local convolution filters does not assure obtaining a sharp image containing a strong blurring component. As a result, existing image deblurring approaches are prone to artifacts problems in a synthesized image when the image containing a strong blurring component. Therefore, to solve the above issues adding an attention mechanism (channel and self-attention) mechanism is essential because they help the network to get information from a clean region when the image containing a strong blurring component. Moreover, adding edge loss to the image deblurring network helps the network to generate an image with a clear edge.

CHAPTER THREE

3. RESEARCH METHODOLOGY

3.1. Overview

This chapter explains the methodology utilized in conducting edge and attention-based image deblurring by GAN. This section describes the dataset, preprocessing, feature extraction algorithm, baseline work, and patchGAN discriminator. Finally, this chapter stated tools and evaluation methods used to access the result of the study. Figure 3.1 describes the overall proposed model process flow.

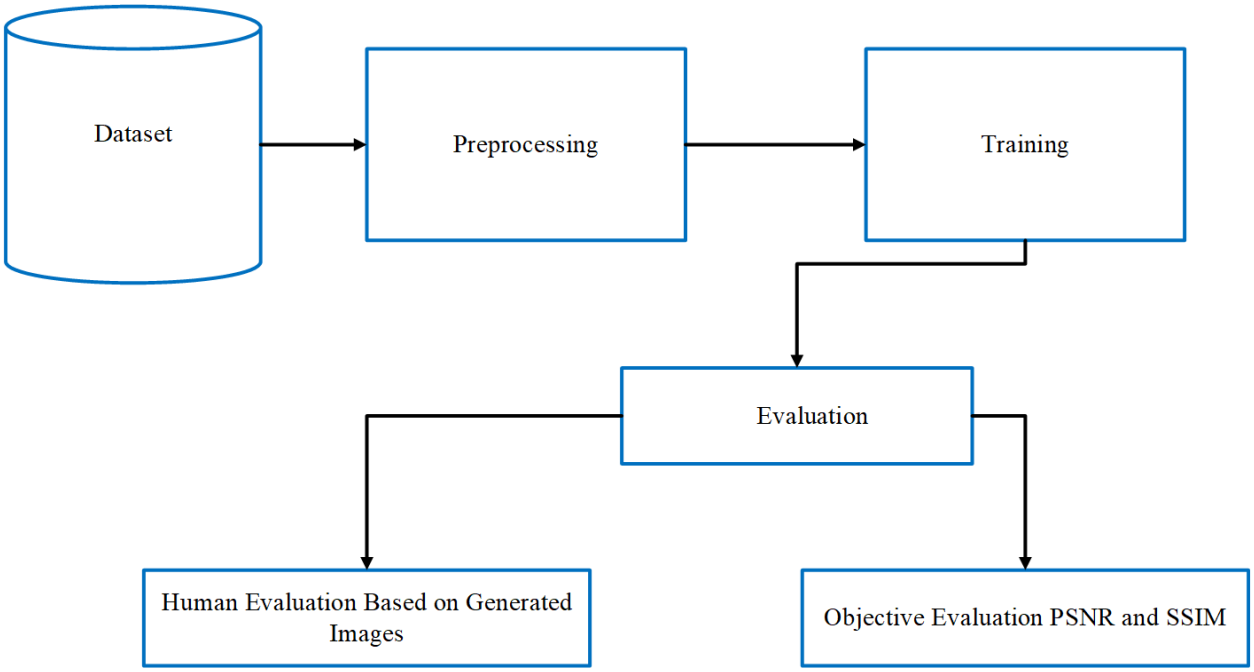


Figure 3. 1 The proposed model process flow

3.2. Dataset

This study uses a deep learning mechanism to solve image deblurring problems in a supervised manner, so data is an important part of the study. GoPro (Nah et al., 2017) [1] is used for both training and testing as in (Kupyn et al., 2019) since it is the widely used dataset in image deblurring and Lai (Lai et al., 2016) dataset can be utilized in this work to conduct a human evaluation. Those are

[1] The GOPRO dataset is publicly available on https://seungjunnah.github.io/Datasets/gopro

freely available online and accessible datasets. GoPro dataset contains a pair of sharp and blurry images and it is used to train and test the model. Lai data set contains only blurry images and it is utilized to conduct a human evaluation.

3.2.1. GoPro Dataset

GoPro dataset (Nah et al., 2017) is a common dataset for motion blurring and contains 3,214 pairs of sharp and blurry images. Instead of employing a kernel to blur a sharp image, a sharp photo is recorded and coordinated over time for blur photogeneration. GOPRO4 camera was used to record video. To synthesize blurs of diverse strength 240 fps videos were taken and averaged between different number (7-13) frames. The sharp image is defined as the medium frame that is utilized to create a blurry image. The blur formation process can be done as:

$$B = g\left(\frac{1}{T}\ E(t)dt\right) \sim g\left(\frac{1}{M}\sum_{i=0}^{M-1} E[i]\ \right) \qquad (3.1)$$

Where $E(t)$ and T reveals sensor signals of the sharp picture at time t and exposure time respectively. Moreover, $E[i]$, M are the ith actual frame signal captured amid the exposure time and the number of examined frames, respectively and g is a camera response function (CRF) that transforms a sharp signal $E(t)$ into an obtained image Ē(t) such that Ē(t) = g (E(t)), or Ē[i] = g (E[i]). Since there is not CRF estimation mechanism available non-uniform motion deblurring becomes troublesome when nonlinear CRF is included (Nah et al., 2017). So to handle the nonlinear CRF problem camera response function were approximated by gamma curve with γ = 2.2 as takes after:

$$g(x) = x^{\gamma} \qquad (3.2)$$

Hence, by redressing the gamma work, the latent frame signal $E[i]$ obtained from the observed image Ē[i] by $E[i] = g^{-1}$ (Ē[i]), and after that generate the comparing blurry picture B by utilizing (3.1).

Figure 3. 2 GoPro dataset sample examples

3.2.2. Lai Dataset

This dataset has a collection of real-world blurry pictures of diverse qualities and resolutions collected in different sorts of scenes. Those real pictures have no clean/sharp partners, making a full-reference quantitative assessment inconceivable. Following (Lai et al., 2016)[2], subjective evaluation (human evaluation) was conducted to compare the deblurring execution on those real blurred pictures as discussed in section 3.7.3.

Figure 3. 3 Sample from Lai test set

[2] The Lai dataset is publicly available on http://vllab.ucmerced.edu/wlai24/cvpr16_deblur_study/

3.2.3. Data Pre-Processing

It is a process of preparing the raw data and making it reasonable for a deep learning model. It is the first and fundamental step before start training the deep learning model. Several methods were used to preprocess data to create a better dataset before training any of the deep learning methods.

A. Normalization: is the process of uniformly scaling image pixels as [0,1] or [-1,1] to ease the training and for faster convergence. In this study, the dataset is normalized between -1 and 1. This means the maximum and minimum of each attribute is between 1 and -1 respectively. To normalize images between -1 and 1 in TensorFlow first the image data type is converted to float. Then the normalized image x can be defined as:

$$x = \left(\frac{y-\mu}{\delta}\right) \qquad (3.3)$$

Where y is the input image that has a value between 0 and 1, μ , and δ are mean and variance respectively. Both mean and variance have 0.5 values.

B. Resize Image: Image resizing is a critical preprocessing step in deep learning that allows adjusting input data to the size of the network. The original image size was 1280x720 and for computation complexity, the input data is resized to 256x256 pixels.

3.3. Development Tools

Development tools can be numerous types of software or hardware that are used to implement and design this work.

3.3.1. Hardware Tools

Table 3. 1 Hardware tools used to do the research

No	Device Name	Purpose
1	RAM	To improve hardware performance
2	Hard Disk	To store datasets and models
3	GPU	To increase the computation and to fasten the training process

3.3.2. Software Tools

Software tools are tools used for writing the code, visualizing results, and documenting the research process for reporting.

Jupyter notebook: is an open-source Python programming language distribution geared toward scientific computing (data science and machine learning applications), intending to simplify package and management. It provides a navigator application to view different settings like Jupiter notebook, spider, vs-code, and modules installed in the environment. Anaconda simplifies package administration and deployment.

TensorFlow: is a machine learning platform that's open source. and will be utilized to build a model and deploy it in client environments like other real-world applications. It has a robust, adaptable ecosystem of tools, libraries, and community assets that permit researchers to thrust the boundaries of machine learning and it simplifies the creation and deployment of ml-powered applications.

Keras: Keras is a powerful and easy-to-use deep learning framework built on top of Tensorflow. It is easy and it is preferable due to its user-friendly modular and extensible work on CPU and GPU though it depends on the task to execute.

OpenCV: is a library mainly aimed to provide shared infrastructure for image processing applications. The library contains over 2500 optimized algorithms including a thorough collection of classic and cutting-edge computer vision algorithms. These algorithms are capable of detecting and identify objects in videos classify actions in videos and find similar images in an image database.

3.4. Baseline Works

To validate the effectiveness of this, work the implemented model is compared to recent models on image deblurring with GAN. Deblurgan-v2 (Kupyn et al., 2019) was used as a baseline in this work since Deblurgan-v2 restores better quality images with small epochs when compared to other existing images deblurring works.

Deblurgan-v2 (Kupyn et al., 2019) uses an FPN to restore the actual image from the blurred input image. This work employs it to makes comparisons to identify whether the quality of the synthesized image is superior.

3.5. Feature Extraction

It is a mechanism for reducing the dimension of input data in which, the raw data is partitioned and decreased to more reasonable groups without losing important information (Kumar & Bhatia,

2014). The foremost critical characteristic of these huge data sets is they have an expansive number of variables. These factors require expensive computing assets to handle. Feature extraction aids to get the finest features from enormous data sets by selecting features that represent the whole dataset. These features are simple to process and describe the original data. Over the last few decades, researchers have proposed a slew of cutting-edge deep CNN architectures for image detection and classification. Inception-ResNet-v2 (Szegedy et al., 2017), Xception (Avery et al., 2014), DenseNet121 (Huang et al., 2017), and EfficientNet (Tan & Le, 2019) are some of the different architectures. Figure 3.3 depicts the top1 accuracy as well as the number of parameters. The number of trainable parameters is shown on the x-axis. In terms of accuracy, EfficientNet outperforms other CNN architectures with fewer parameters. Based on the perception made above, EfficientNet-B7 is used as the backbone in the generator for feature extraction in this study to improve the network performance with small computational complexity.

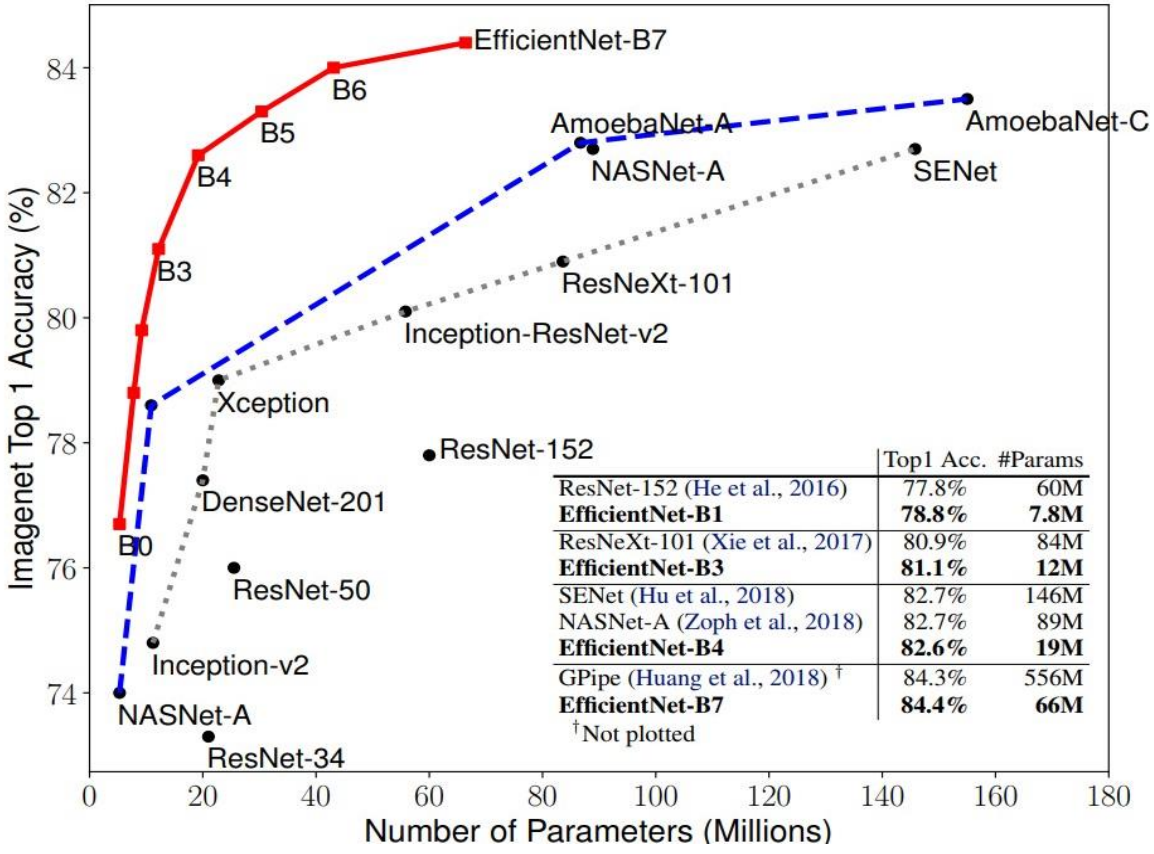


Figure 3. 4 ImageNet Accuracy vs Model Size in parameter

Source: Adapted from (Tan & Le, 2019)

EfficientNet-B7 was made up of seven blocks and each block further has a different number of sub-blocks. All these sub-blocks can be made from five different modules (*Complete Architectural Details of All EfficientNet Models | by Vardan Agarwal | Towards Data Science*, n.d.). Module one is utilized as the starting point for sub-blocks and contains DepthwiseConv2D, BatchNormalization, and Activation, module two contains DepthwiseConv2D, BatchNormalization, Activation, ZeroPadding, DepthwiseConv2D, BatchNormalization, Activation, and it is used as an initial point for the first sub-block of all the main blocks except the first block, module three contains GlobalAveragePooling2D, Rescaling, Conv2D, Conv2D and its is used to connect all subblocks by skip connection, module four is made up of element-wise multiplication, Conv2D, BachNormalization and utilized for joining the skip connection with the first sub-blocks and module five contains element-wise multiplication, Conv2D, BatchNormalization, Dropout and it is utilized to combine previous subblocks.

3.6. PatchGAN Discriminator

When dealing with image generation issues, using only L2 and L1 loss does not promote high-quality image generation because it produces blurry results due to failure to capture the high frequencies. So, is adequate to concentrate on high frequencies to generate high-quality images. PatchGAN discriminator is a mechanism that solves this issue by capturing high frequency to generate high-quality images. PatchGAN discriminator is designed by Philip Isola et.al (Isola et al., 2017) and it penalizes areas at the scale of patches. This discriminator attempts to decide whether each of an image's NxN patches is fake or real. Where N can be much smaller than the image's full size and help to produce high-quality results since it operates at the local image patch. This makes PatchGAN has small parameters, applied to images of any size, and run faster. This type of discriminator represents the image as a Markov random field with pixels separated by more than a patch diameter assumed to be independent. Recently, Jo et.al (Jo & Park, 2019; Ozcinar et al., 2019) use a patchGAN discriminator and outperforms prior works in terms of image quality. Based on this perception the PatchGAN discriminator was used in this study.

3.7. Evaluation Method

Any handling applied to an image may cause an important loss of data or quality(Horé & Ziou, 2010). There are different metrics used to assess the quality of an image. The metrics are classified as subjective or objective. Subjective methods depend on human judgment, and there are no explicit criteria and rules. Objective methods are depending on explicit numerical criteria in terms

of statical parameters. The objective evaluation method was used to describe the execution of the image deblurring model on the test dataset. The dataset is divided into test and training sets, and the algorithm is evaluated on the test set. One major issue with GANs is that there is no reliable way to go beyond visual inspection so to address this problem human evaluation is used in this work. The following section provides a subjective (human evaluation) analysis as well as an objective metric (PSNR, SSIM) to assess this work.

3.7.1. PSNR (Peak Signal to Noise Ratio)

It is the ratio of a signal's maximum possible value to the ability to distort noise that impacts the representation's quality (Isa et al., 2015). The PSNR will be used to evaluate image quality after reconstruction, with a higher PSNR indicating a better reconstruction. PSNR based on MSE and MSE is calculated as :

$$\text{MSE} = \frac{1}{MxN} \sum_{i=1\ j=1}^{i=M\ j=N} (Y(ij) - X(ij))^2 \qquad (3.4)$$

where X and Y show the actual and synthesized images of size MxN respectively.

PSNR is expressed in dB and formulated as:

$$\text{PSNR} = 20 * \log10 \left[\frac{\text{K}}{\sqrt[2]{MSE}}\right] \qquad (3.5)$$

where K is the maximum signal intensities.

3.7.2. SSIM (Structure Similarity Index Measure)

It is an algorithm for assessing the perceptual quality of natural images (Bovik et al., 2005). The SSIM compares a test output image X to an original image Y to determine their visual similarity based on contrast, luminance, and structural terms.

Luminance: it's measured with the aid of using averaging all pixel values of the image. It is characterized by a function, l(x, y) which is appeared below.

$$l(x, y) = \frac{2\mu_s\mu_b + C_1}{\mu_s^2 + \mu_b^2 + C_1} \qquad (3.6)$$

Where, C_1, μ_s, μ_b are constant protects denominator form zero, mean of real and restored images respectively.

Contrast: It is obtained by taking the standard deviation of the pixel values. It is indicated by σ(sigma) and represented by:

$$c(x,y) = \frac{2\sigma_s \sigma_b + C_2}{\sigma_s^2 + \sigma_b^2 + C_2} \quad (3.7)$$

Where, C_2, σ_s, σ_b are constant protects denominator form zero, the standard deviation of real and restored images respectively.

Structure: It is denoted by the function s(x, y) which is described below.

$$s(x,y) = \frac{\sigma_{sb} + C_3}{\sigma_s \sigma_b + C_3} \quad (3.7)$$

Where, C_3, σ, σ_{sb} are constant protects denominator form zero, standard deviation and σ_{sb} calculated as:

$$\sigma_{sb} = \frac{1}{N-1} \sum_{i=0}^{N} \left(x_i - \mu_x\right)\left(y_i - \mu_y\right) \quad (3.8)$$

Finally, the SSIM score is obtained by:

$$\mathbf{SSIM}(\mathbf{x}, \mathbf{y}) = [l(x,y)]^{\alpha} * [c(x,y)]^{\beta} * [s(x,y)]^{\gamma} \quad (3.9)$$

Where $\alpha > 0$, $\beta > 0$ and $\gamma > 0$ are parameters used to adjust the three components and l, c and s are luminance, contrast, and structure respectively.

3.7.3. Human Evaluation Study

Comparing the abilities of various models is difficult in GAN because lacks an objective function. This means that there isn't a widely used method for evaluating a specific GAN generator model. The objective assessment models of GAN generators are still a work in progress. Regardless of the fact, several measures have been implemented there is still no agreement on which measure best captures model strengths, and limitations should be utilized to compare models reasonably. To tackle this issue GAN model performance was evaluated by a human. This evaluation method employs 15 to 25 volunteers who assess whether the given image is better than baseline work after seeing sharp images restored by this work and baseline work. The average score value is obtained using the number of figure entities. The higher the human evaluation score means, the better the execution of the network.

CHAPTER FOUR

4. PROPOSED EAIDGAN ARCHITECTURE

4.1. Introduction

This chapter describes in detail the proposed architecture of edge and attention-based image deblurring by GAN. Ideally, this chapter can be organized into three main parts; the first part describes the proposed model architecture; the second part gives an overview of model learning functions. The last part explains Pseudocode utilized to train the model.

4.2. Proposed Architecture

The model architecture has been adopted from the architecture defined in "Feature Pyramid Networks for Object Detection" (Lin et al., 2017) for learning mapping functions between blurry and sharp images. To handle large size blur kernel residual blocks and self-attention layer (H. Zhang et al., 2019) were added as bottleneck layer, and channel-attention layer was used after each upsampling block to model the channel correlations. To clearly understand the architecture, let's make a short introduction to the following topics.

4.2.1. Feature Pyramid Network (FPN)

The FPN module was originally proposed for detecting objects (Lin et al., 2017). It generates multi-layer feature maps which encode diverse semantics and contain quality information. FPN includes top-down and bottom-up pathways. The bottom-up pathway is feedforward ConvNet for feature extraction, throughout this spatial resolution is downscaled by half, but more important information is extracted and compressed. The top-down pathway remakes higher spatial resolution from the semantically wealth layers (Lin et al., 2017). To take advantage of sematic information on different levels those features are resized to ¼ of input shape and concatenated into one tensor as shown in figure 4.1. To get the original image size and reduce artifacts two upsampling block and convolution layers were added after the concatenation layer. To seek after the state-of-the-art deblurring quality and minimize computational complexity pre-trained EfficientNet-B7 (Tan & Le, 2019) was utilized as a feature extractor in the generator network of GAN.

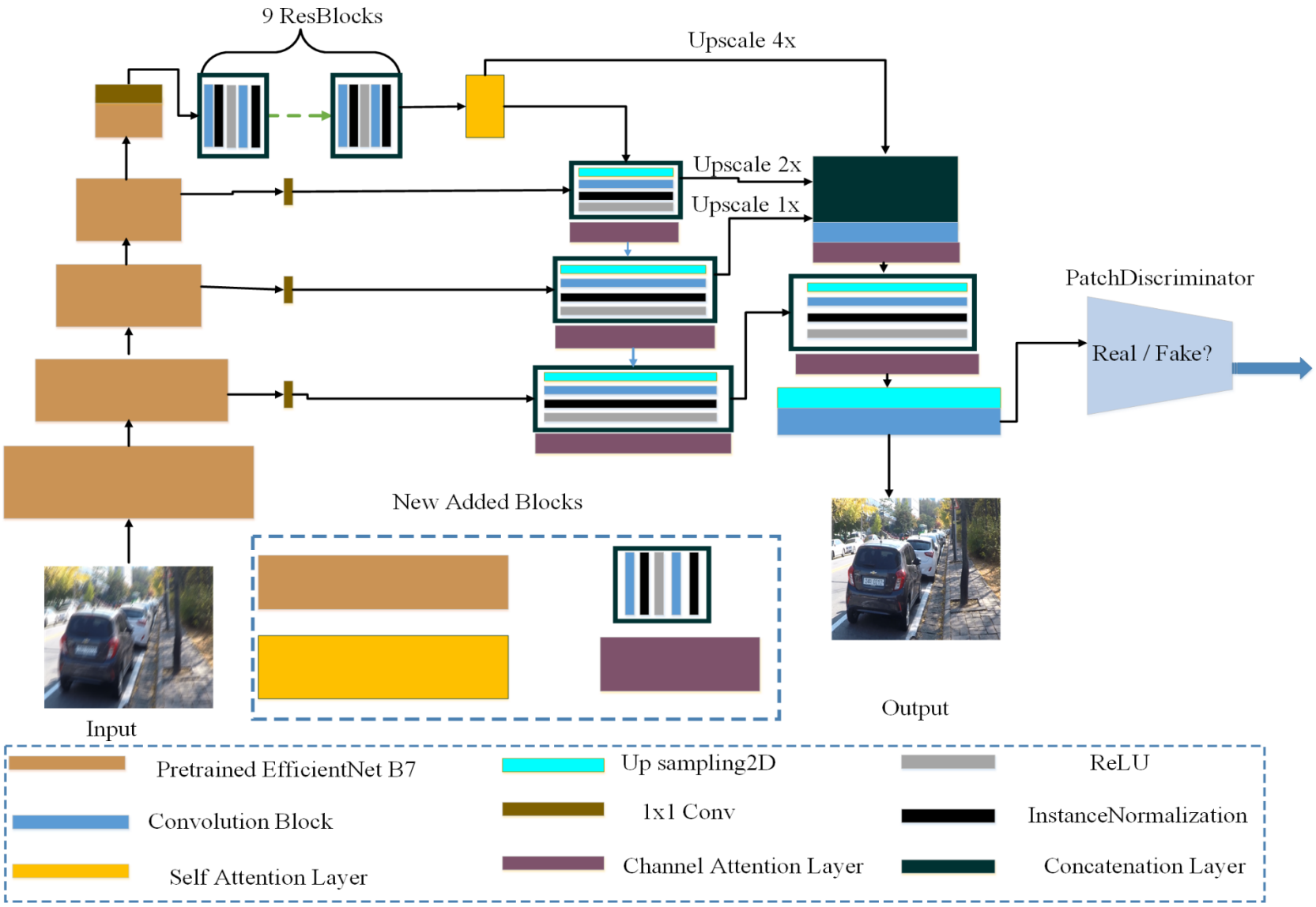


Figure 4. 1 Proposed EAIDGAN architecture

4.2.2. Bottom-up Pathway

The bottom-up pathway is feedforward ConvNet for feature extraction, throughout this resolution is downscaled by half, but more important information is extracted and compressed. To extract more important information pre-trained EfficientNet-B7 (Tan & Le, 2019) is used in bottom-up pathways since EfficientNet-B7 has fewer parameter and outperform other feature extraction net works as described in section 3.5. The output of the high-level feature at "block6a_expand_activa tion", "block4a_expand_activation", "block3a_expand_activation", and "block2a_expand_activa tion" are used as feature extractor. Each pre-trained block takes input from the previous block an d extracts important information by reducing the resolution by half and passes to the block next t o it as shown in figure 4.2. Finally, the channel of the last pre-trained block was reduced to 256 by 1x1 convolution and passed as input to the residual block.

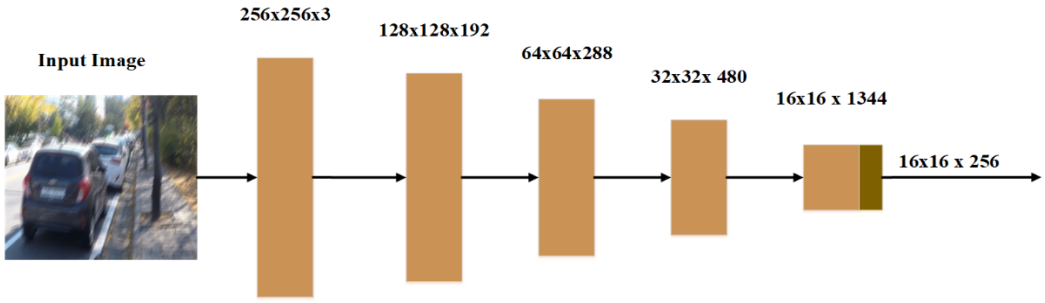


Figure 4. 2 Bottom-up pathway

4.2.3. Residual Blocks

The residual block is a shortcut connection block, which learns the residual function regarding the input of the layer. Due to computationally less expensive and shortcut connection residual blocks are used to train very deep networks without vanishing gradient problems (He et al., 2016). To handle a large blur, the size of the receptive field is very important. Xintao Wang et.al (X. Wang et al., 2019) applied residual blocks to super-resolution problems and improve the execution of the network. Based on this to handle the large blur size and to improve performance nine residual blocks were used in this work. Residual blocks contain a convolution, instance normalization, ReLU, Dropout with a probability of 0.5, convolution, and ReLU activation respectively as appeared in figure 4.3. This block takes the bottom-up pathway output as input, extracts more information, and passes the output to the self-attention layer.

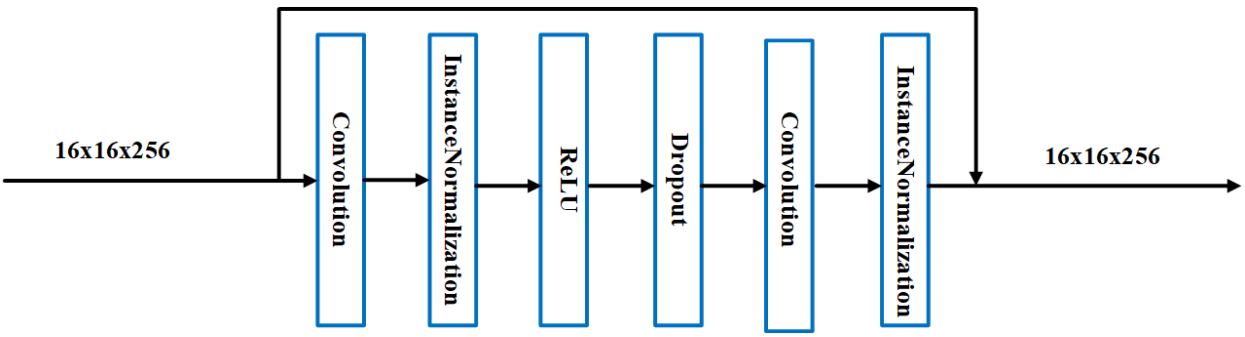


Figure 4. 3 Residual blocks

4.2.4. Self-Attention Layer

Recently, attention mechanisms are popular and most useful algorithms in deep learning. They are motivated by how humans pay attention to different parts of an image. In computer vision, the attention mechanism makes the network focus an important part of the image during image generation and this makes the network synthesize high-quality images. For example, self-attention GAN (SAGAN) improved the best-reported baseline inception result from 36.8 to 52.52 and reduced Fréchet inception distance from 27.6 to 18.65, which is significantly better than the previous image synthesis work (H. Zhang et al., 2019). Yuan et.al (Yuan et al., 2020) uses the self-attention layer into the up-sampling part of the generator to tackle the issue that the restricted size of the convolution kernel causes the network to be unable to learn a wide range of global dependencies and outperforms previous work in terms of quality. Based on this observation self-attentions were added to the generator model in GAN as the bottleneck layer next to residual blocks. This layer takes the output of the residual block as input and models the correlation or dependency between pixels to help the network concentrate on the important part of the image and passes the output to the top-down pathways. SAGAN mathematical expression and diagram were discussed in section 2.2.3.

4.2.4. Top-down Pathway

The top-down pathway takes the output of the self-attention layer and remakes higher spatial resolution from the semantically wealth layers through the up-sampling block. As input goes down within the top-down pathway the spatial size of an image is doubled and the channel is reduced. Each up-sampling block contains upsampling2D, Conv2D, Instancenormalization, ReLU, and channel-wise attention layers respectively as appeared in figure 4.4. To improve the feature maps of top-down pathways lateral connection was used as skip connection between the same spatial size of bottom-up and top-down pathways since bottom-up activation is more precisely localized

(Lin et al., 2017). Each lateral connection performs 1x1 convolution to minimize the number of channels. To take advantage of feature fusion features of semantic information on different locations are resized to ¼ of input shape (64x64) and concatenated to one tensor and then followed by two up-sampling blocks to get the original image size as appeared in figure 4.4.

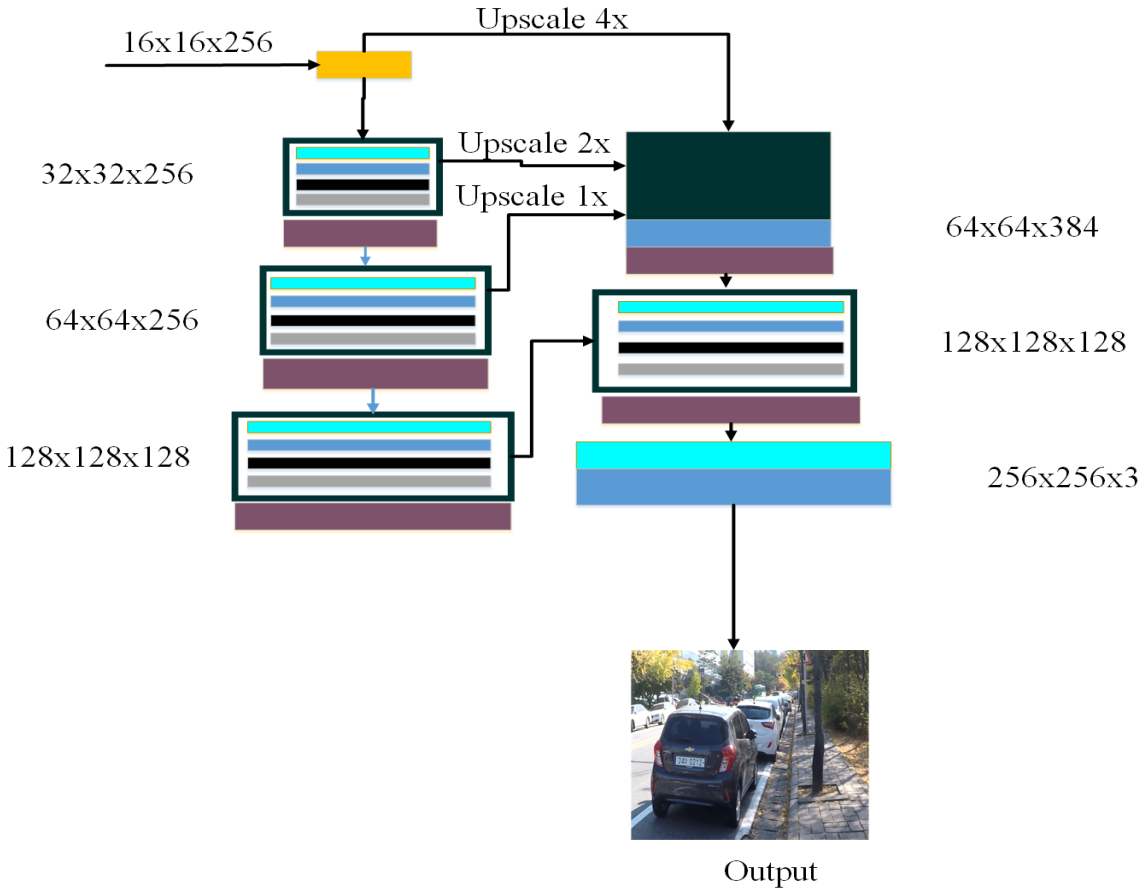


Figure 4. 4 Top-down pathway

4.2.5. Channel Attention

Channel attention is a mechanism that makes the network focus on more important futures and exploits the relationship between feature channels (Hu et al., 2020). How to create diverse considerations for each channel-wise feature could be a key step. Because convolutional layer filters operate on the small receptive field it is difficult to obtain the sharp image from a blurry image blurred by heavy camera shake and object movement simultaneously happen, so the global spatial relationship too ought to be considered. For example, in the image containing a strong

blurring component exact discrimination utilizing any conceivable strategy may require relevant data from clean regions.

Based on these examinations, the channel-based global spatial information takes into a channel descriptor by utilizing global average pooling. As appeared in figure 4.5. let M = [m1, · · ·, mc, · · ·, mD] be an input, which has D feature maps with size WxH. The channel-based statistic z ∈ R^C can be obtained by aggregating M through spatial dimensions H ×W. After that, the c-th member of z is obtained as:

$$z_c = F_{GP}(y_c) = \frac{1}{HxW}\sum_{i=1}^{H}\sum_{j=1}^{W} y_c(i,j) \qquad (4.1)$$

Where $y_c(i,j)$ is the value at position (i, j) of the c-th feature y_c. $F_{GP}(.)$ shows the global average pooling function. Such channel measurements can be seen as a set of the nearby descriptors, whose measurements contribute to express the total image.

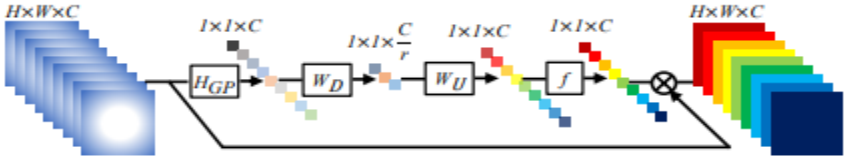


Figure 4. 5 Channel attention (CA). ⊗ shows element-wise product

Source: Adapted from (Y. Zhang et al., 2018)

As discussed in (Hu et al., 2020), to capture channel-based relationships from the shrunk information by global average pooling gating mechanism is introduced. The gating mechanism should fulfill two criteria. To begin with, it must be able to memorize nonlinear intuition between channels. Moment, as numerous channel-based features can be emphasized as contradicted to one-hot actuation, it must learn a non-mutually-exclusive relationship as in 4.2.

$$s = f(WU\ \delta(WD^z) \qquad (4.2)$$

where $f(.)$ and $\delta(.)$ show the sigmoid gating and ReLU function, respectively. WD indicates the weight set of a Conv layer, which downscales the channel with a reduction rate r. After being activated by ReLU, the low-dimension flag is at that point expanded with proportion r by a channel-upscaling layer, whose weight set is WU. Then the final channel statistics s obtained by:

$$\hat{Z} = z_c.y_c \qquad (4.3)$$

Where y_c and $z_c$are input feature map and scaling factor in the c-th channel.

Yulun Zhang et.al (Szegedy et al., 2017) uses a channel attention mechanism in image super-resolution and obtain the high-quality image from low-resolution images and outperform other previous works. Based on this observation channel attention was used in this study after each top-down pathway. It takes the output of the top-down pathway and filter the most important channel and passes the result to the next top-down block next to it as shown in figure 4.1

4.3. Model Learning Functions

It is well known that planning a viable learning function is exceptionally key for yielding wonderful results. The loss function is utilized to assess the gap between the synthesized image and the real image which is the optimization protest of the GAN. The littler the loss function value, the littler the gap between the synthesized image and the real image, and the superior the reconstruction impact. A sensible learning function can give exact gradient information for network training, subsequently improve reconstruction execution (performance).

To fix the research question, this work adds a learning function to improve the quality of the restored image. Since this work follows the GAN architecture, the Least square GAN loss, called ℓ**LSGAN** is used in this work to train the generator and discriminator. And the feature loss ℓ**feature,** Content loss ℓ**Content,** and edge loss ℓ**edge** are too utilized. The total loss function of this model is as follows:

$$\ell\text{total} = \ell\text{LSGAN} + \gamma\ell\text{feature} + \alpha\ell\text{Content} + \beta\ell\text{edge} \quad (4.4)$$

Where γ, α, β, are used hyperparameters weights utilized to adjust weights.

To fully understand the model learning function let's look in detail at all loss functions.

4.3.1. Least Squares loss (LS)

Instead of vanilla GAN loss, LS loss is introduced in this work to train generator and discriminator simultaneously. As discussed in Section 2.2.3 the LS loss function draws fake samples to the decision boundary, by penalizing samples that are on the right-side decision boundary but, far from the decision boundary and this helps the network to generate high-quality images. The Least Squares loss is obtained as:

$$\ell\text{LSGANG} = \frac{1}{2}E(z) \sim \text{Pz(z)}\left[(\text{D(G(z))} - \text{c})^{2}\right] \quad (4.5)$$

Where c shows the value that the generator used to trick the discriminator and Pz(z) is the probability of fake data distribution.

4.3.2. Content Loss

Content loss guarantees that the network holds the color structure of the image. Content loss utilizes mean-square error (MSE) or mean absolute error (MAE) loss to access the pixel-based content closeness. But both losses causing excessively smooth results (S. Zhang et al., 2020). In this work, Charbonnier loss is presented as a content loss to preserve color details (Lai et al., 2017). As appeared in (Lai et al., 2017), it gives pixel-space regularization for loss optimization and contributes to subjective enhancement. The content loss is obtained as:

$$\ell\text{Content} = \frac{1}{\mathrm{k}}\sum_{i=1}^{k} \sqrt{\left(G_{\theta G}\left(I_i^{bl}\right) - I_i^{sh}\right)^2 + \epsilon^2} \qquad (4.6)$$

Where $G_{\theta G}\left(I_t^{bl}\right)$ represents the restored image, $I_t^{sh}\ \ i = 1, \ldots, k$ real image responding with a blurry image $I_t^{bl}\ i = 1, \ldots, k$ and θ represents the parameters of the generator. ϵ is a penalty value near to 0, which shows the impact of the Charbonnier penalty.

4.3.3. Feature Loss

Feature loss measures high-level feature and semantic contrasts between restored and ground truth images (Johnson et al., 2016) to reduce feature disappearance problems during training. It utilizes feature maps extracted after activation in feature loss. However, wang et.al (Rakotonirina & Rasoanaivo, 2020) have illustrated that utilizing feature maps before activation can be creating more exact texture details. Thus, following (Rakotonirina & Rasoanaivo, 2020), the high-level feature of the layer "Conv4-3" of the pre-trained VGG19 network was utilized to preserve the feature structure of an image.

$$\ell\text{feature} = \left\|\Omega\left(I^{sh}\right) - \Omega(G_{\theta G}\left(I^{bl}\right))\right\|_2^2 \qquad (4.7)$$

Where $G_{\theta G}\ (I^{bl})$ and $\left(I^{sh}\right)$ are the restored image and true real image, respectively and Ω shows feature maps from the pre-trained vgg-19 model.

4.3.4. Edge Loss

Having sharp edges is an instinctive judgment to demonstrate the image clarity of an image. Sharp images have a clear edge while the blurry image has not a clear edge because the shape of the items in the image was distorted by motion blur as appeared in figure 4.6. So to obtain a sharp image with a clear edge from a blurry image it is better to measure the edge features divergence

between the generated image and the ground truth image. This result is used as generator loss and makes the generator synthesize sharp images close to the ground truth image. Based on this observation Sobel edge detection is introduced in this work since it is based on the first-order derivative and requires a small computational cost. Edge loss can be calculated with the outlined Sobel convolution (Chaple & Daruwala, 2014) layer by using Sobel kernel operators. It utilizes the convolution operation and kernels to extract edges map. The algorithm uses two kernels to extract edge information along the x-direction and y-direction. The two kernels are convolved with each pixel within the image to recognize the districts where the change (gradient) is maximized in size within the x and y magnitude. The point where the gradient change is understood as an edge in the Sobel algorithm. The edge loss ℓedge is obtained as:

$$\ell\text{edge} = \frac{1}{WH}\sum_{x=1}^{W}\sum_{x=1}^{H} \left(\left\|\nabla_h(sh)_{x,y} - \nabla_h\left(G(blr)\right)_{x,y}\right\|_1 + \left\|\nabla_v(sh)_{x,y} - \nabla_v(G(blr))_{x,y}\right\|\right)_1 \quad (4.8)$$

Where ∇_h and ∇_v are the Sobel operators along the horizontal and the vertical directions respectively, H and W are the height and width of inputs respectively and (sh) and $G(blr)$ are ground truth and generated image respectively.

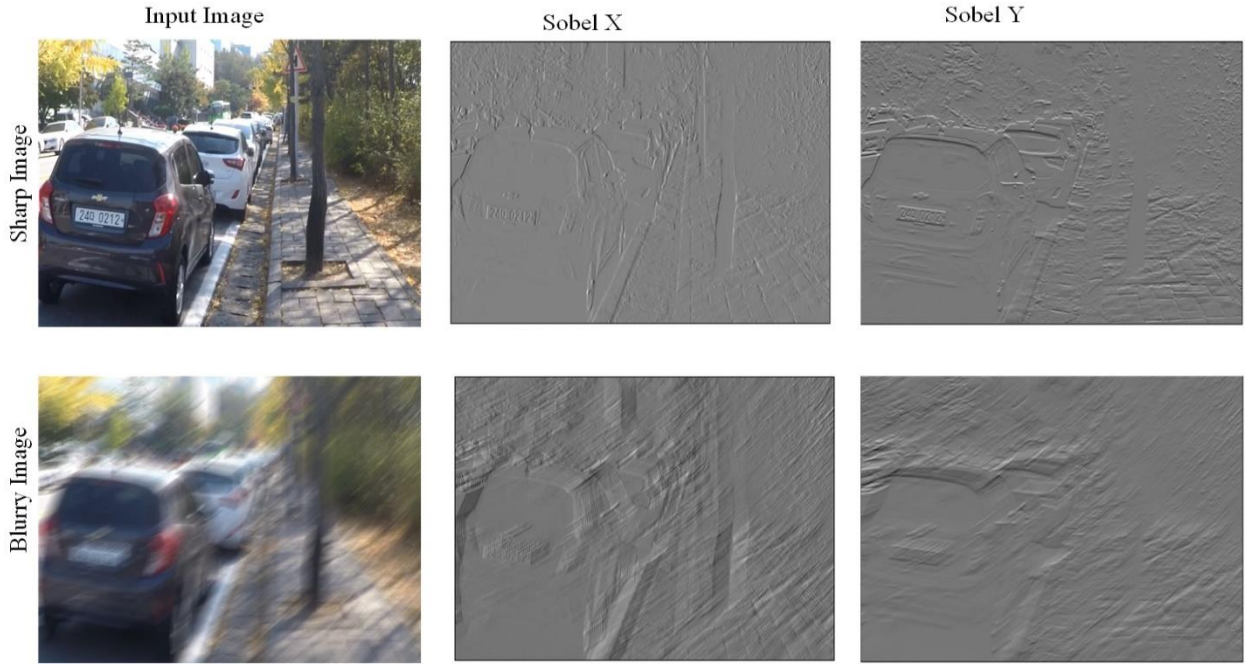


Figure 4. 6 Sharp and blurry image edge features extracted by Sobel operator

4.4. Patch Discriminator

To improve quality Patch GAN discriminator (Isola et al., 2017) is adopted in this work. As stated in section 3.6 PatchGAN discriminator makes the network produce high-quality results since it penalizes only structure at the local scale of patches. The discriminator architecture and the output stay the same with PatchGAN (Isola et al., 2017). As depicted in figure 4.2 discriminator architecture contains a series of Convolution, Instancenormalization, and LeakyReLU layers with alpha 0.2 except the first and the last layer. Like a generator, Least Squares loss is used as discriminator loss. The Least Squares loss of the discriminator is mathematically expressed as follows:

$$\ell \text{LSGAND} = \frac{1}{2} E(x) \sim \text{Pdata(x)}\left[(\text{D(x)} - \text{b})^{2}\right]$$

$$+\frac{1}{2} E(z) \sim \text{Pz(z)}\left[(\text{D(G(z))} - \text{a})^{2}\right] \qquad (4.9)$$

Where, a and b are the labels for fake data and real data, respectively, and Pdata, Pg shows real data distribution and generated data distribution respectively.

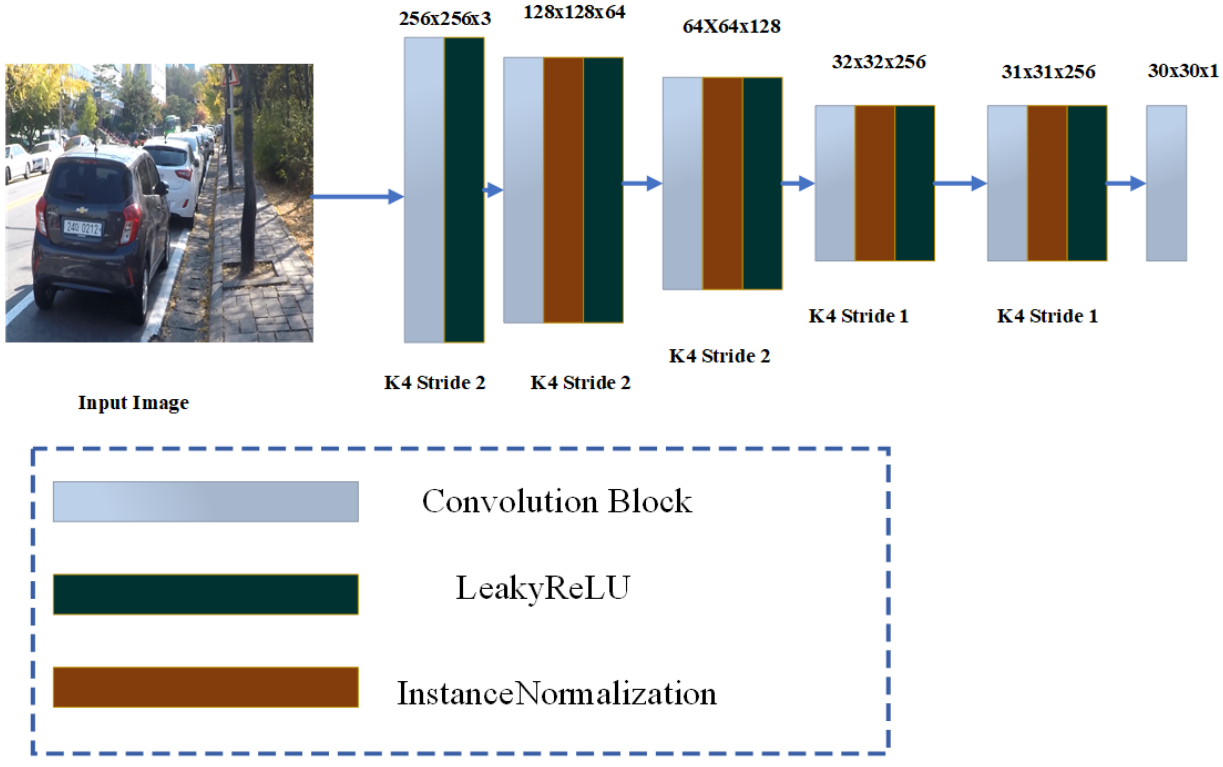


Figure 4. 7 Patch discriminator architecture

4.5. Training Pseudocode

This section describes the training algorithms used in this study. The procedure in Table 4.1 was used to train the channel and self-attention-based image deblurring by the GAN model in the same manner with which most GAN are trained. The entire configuration and training process is aimed at reducing the dissimilarity between the Ground truth and the restored image.

Algorithm 1: Proposed architecture with total loss and PatchGAN discriminator

Input: batches of blurry images

Output: restored image

1. Load batches of data from the dataset into memory

2. Give the batch of input to the generator network

3. Train discriminator:

    3.1. Restore the sharp image from the blurry batch of input

    3.2. Calculate discriminator least square loss by using restored and ground truth image

    3.3. Update discriminator parameter to minimize classification loss

4. Train generator:

    4.1. Restore the sharp image from the blurry batch of input

    4.2. Least square loss of restored image is computed

    4.3. Content loss is calculated with the original sharp and the restored images

    4.4. Feature loss is computed with the original sharp and the restored images

    4.5. **Edge loss is computed with the original sharp and the restored images**

    4.6. Update generator parameters based on least square, content, feature, and edge loss to minimize reconstruction loss

The bold indicates the added learning functions.

CHAPTER FIVE

5. IMPLEMENTATION DETAILS

5.1. Overview

This chapter describes the implementation of the proposed model. The working environment and edge and attention-based image deblurring by GAN implementation are discussed. In this section, important portions of code were displayed while the imperative detail codes were outlined in the appendix section.

5.2. Working Environment

This section describes the hardware specification that has been used to implement the conditional GAN experiment.

- Desktop Computer: used for implementing and training the edge and attention-based image deblurring by GAN model.
  - Model: DELL OptiPlex 9020
  - Operating system: Windows 10
  - Processor: Intel ® Core™ i5-4580 CPU @ 3.29GHz x 4
  - Graphics Card Upgrade: GeForce RTX 2070 Super 8 GB RAM
  - Memory Size: 12.00 GB
  - System Type: 64-bit Operating System, x64-based Processor
- Laptop computer: The laptop computer is utilized for creating a network architecture.
  - Model:3168NGW
  - Operating system: windows 10 pro
  - Processor: Intel ® Core™ i3-7020U CPU @ 2.30 GHz
  - Memory Size: 4.00 GB
  - System Type: 64-bit Operating System, x64-based Processor

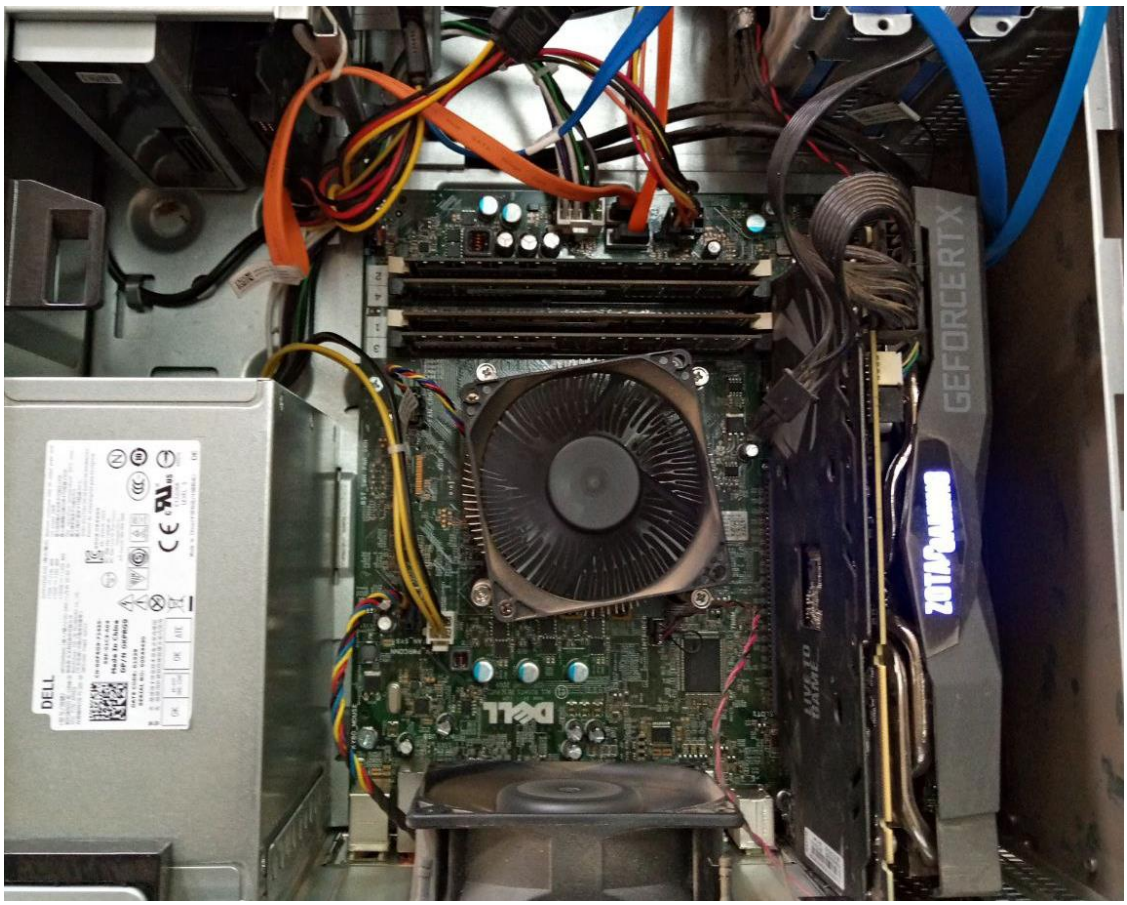


Figure 5. 1 GPU configuration as components in other systems

5.3. Setup Environments

5.3.1. Integrated Development Environments and Editors

Jupyter Notebook: is the most popular Block-wise programming IDE used to edit python code and to visualize diagrams. This thesis work uses Jupiter notebook 6.0.0.

Colab: is Online Jupyter Notebook provides several utilities for running codes General processing Unit (GPU), Tensor Processing Unit (TPU), and Random-Access Memory.

Visual-Studio-Code: is a popular source code editor IDE that is easily configured and used to edit different programming languages such as c++, python, java, and other programming languages. In addition to Jupiter notebook, Visual-Studio-Code version 1.56.2 was installed for implementation of this work.

5.3.2. Programming Language and Libraries Used

- Python 3.8.3: is used in this work due to many packages and libraries for scientific computing and large community support are available on python.
- Keras 2.4.3: deep learning architecture designing framework which used to apply different operations such as convolution, pooling, and other deep learning processes and to load the pre-trained model.
- TensorFlow 2.4.1: is used for computations and to connect the retrained model to the graphs.

5.4. Training Procedure

5.4.1. EAIDGAN Implementation

As described in chapter four EAIDGAN is based on an FPN to get sharp images from the blurry image. The EAIDGAN consists of two top-down and bottom-up pathways to extract features and reconstruct the image. The bottom-up pathway is downscale the size of an image and extracts more semantic important information. The bottom-up pathway utilizes EfficientNet-B7 as a feature extractor to extract key features of the input image. The bottom-up pathway is followed by 1x1 convolution which is used as a skip connection between top-down and bottom-up and to reduce the channel of the pre-trained model for computational complexity as shown in code snippet 5.1.

```
from tensorflow.keras.applications import EfficientNetB7
def generator():
    i_shape=Input(shape=(256,256,3))
    modl=EfficientNetB7(weights='imagenet',include_top=False,
input_tensor=i_shape)
    e3=modl.layers[-253].output  #16x16
    e2=modl.layers[-550].output  #32x32
    e1=modl.layers[-654].output  #64x64
    e0=modl.layers[-758].output  #128x128
    #lateral connections
    lateral_0 = Conv2D(128, kernel_size=1,strides=1,padding='same')(e0)

    lateral_1 = Conv2D(256, kernel_size=1,strides=1,padding='same')(e1)

    lateral_2 = Conv2D(256, kernel_size=1,strides=1,padding='same')(e2)

    lateral_3 = Conv2D(256, kernel_size=1,strides=1,padding='same')(e3)
```

Code snippet 5. 1 Bottom-up Pathway Implementation

The output of the bottom-up pathway passes through the bottleneck layer (residual blocks and self-attention layer) to increase receptive field and to learn dependencies between pixels as

shown in the code snippet 5.2. Residual blocks are implemented by the function called resnet_block.

```
def resnet_block(inputs, filters):
    _res = inputs
    x =  tf.pad(inputs, [[0,0],[1,1],[1,1],[0,0]], mode = 'REFLECT')
    x = Conv2D(filters=filters,kernel_size=3,strides=1,padding='Valid')(x)


    x =  InstanceNormalization(axis=3)(x)
    x =  ReLU()(x)
    x = Dropout(0.5)(x)

    x = tf.pad(x, [[0,0],[1,1],[1,1],[0,0]], mode = 'REFLECT')
    x = Conv2D(filters=filters,kernel_size=3,strides=1,padding='Valid')(x)
    x = InstanceNormalization(axis=3)(x)
    skip = _res+x
    return skip
map_3=lateral_3

#bottleneck layer
b1 =  resnet_block(map_3,256)
b2 =  resnet_block(b1,256)
b3 =  resnet_block(b2,256)
b4 =  resnet_block(b3,256)
b5 =  resnet_block(b4,256)
b6 =  resnet_block(b5,256)
b7 =  resnet_block(b6,256)
b8 =  resnet_block(b7,256)
b9 =  resnet_block(b8,256)
self_attention = SelfAttention()(b9)
```

Code snippet 5. 2 Bottleneck Layers Implementation

The top-down pathway takes self-attention output as input and remakes higher spatial resolution from the semantically wealth layers. The top-down pathway contains a series of upsampling2D, Conv2D, Instancenormalization, ReLU, and channel attention layers. To add resolution details and localize object lateral skip connections was used between top-down and bottom-up pathways. Since information on different levels of image, regions contain different important semantic information features of different levels are concatenated into one tensor by resizing to 64x64 as shown in the code snippet 5.3.

```
    map_2=lateral_2+UpSampling2D((2, 2))(channe_att)
    map_2=conv_bn_relu(map_2) #32*32
    channe_att = channel_attention (map_2,256) #channel attention layer

    map_1=lateral_1+UpSampling2D((2, 2))(channe_att) #64*64
    map_1=conv_bn_relu(map_1) #64  #conv2D,InstanceNormalization,Relu
layer
```

```
map_0=lateral_0

#upsample images to 64x64
map_1 = feature_pyramid_head(map_1)
map_1 = upsample_iterations(map_1, scale_iterations=0)

map_2 = feature_pyramid_head(map_2)
map_2 = upsample_iterations(map_2, scale_iterations=1)

map_3 = feature_pyramid_head(map_3)
map_3 = upsample_iterations(map_3, scale_iterations=2)
#concatenation layer
xr = tf.keras.layers.Concatenate()([map_3, map_2, map_1])
x = smooth_1(xr)
```

Code snippet 5. 3 Top-down and Future Fusion Implementation

As described in section 4.2.1 after sematic information resized to ¼ of the input shape (64x64) and concatenated to one tensor, the concatenation layer was followed by two upsampling blocks to get the original image size as shown in the code snippet. Finally,tanh activation function was used as an activation function in the last convolution layer as shown in the code snippet 5.4.

```
channe_att = channel_attention(x,128) #channel attention layer

x = UpSampling2D((2, 2))(channe_att)

x = (x + map_0)
x = smooth_2(x)

channe_att = channel_attention (x,64)

x = UpSampling2D((2, 2))(channe_att)

final_conv = Conv2D(3, kernel_size=3, padding="same")(x)
residual = tf.tanh(final_conv)
```

Code snippet 5. 4 Top-down Pathway Implementation

5.4.2. Feature Loss Implementation

As described in section 4.3.3 feature loss measures high-level feature and semantic contrasts between restored and ground truth images to reduce feature disappearance problems during training. To do so, the pre-trained VGG19 model is imported. Since the goal is to extract the feature map of the input image, the layers after “block4_conv3” are removed, as shown in the code snip. 5.5. Using a pre-trained VGG19 model, the new Tensorflow model has been created.

```
from keras.applications.vgg19 import VGG19

vgg=VGG19(include_top=False, weights='imagenet',input_shape=( 256,256,3))

vgg.trainable = False
loss_model = KM.Model(inputs=vgg.input,outputs=vgg.get_layer
('block4_conv3').output)
```

Code snippet 5. 5 Pre-Trained VGG19 Model

To measure and return the feature value of the restored and real image a function called perceptual_loss is defined. First, the perceptual_loss function extracts the feature map of restored (generated fake image) and real image. After that feature loss between real and restored (fake) image pairs can be computed by L2 distance between the feature map of the real and restored image. Then the loss has been used to modify network weight. As appeared in code snippet 5.6.

```
#computes feature loss for sharp and generated images
@tf.function()
def perceptual_loss(fake,real):

  vggX = loss_model(real)
  vggY = loss_model(fake)

  return tf.reduce_mean(tf.square(vggY-vggX))
```

Code snippet 5. 6 Feature Loss Fonction Implementation

5.4.3. Patch Discriminator Implementation

To improve generator training patch discriminator was introduced in this thesis work. The discriminator architecture is similar to (Isola et al., 2017) discriminator. It takes images as input to discriminate whether the images are real or fake at the patch level as described in section 3.6. This helps the network to strictly focus on the local region of the image as shown in code snippet 5.7.

```
def patch_discriminator():

    input_shape=KL.Input(shape=(256,256,3))
    kw = 4
    pad = int(np.ceil((kw-1)/2))
    c1 =tf.pad(input_shape,[[0,0],[pad,pad],[pad,pad],
[0,0]],mode='REFLECT')
    c1 = KL.Conv2D(64, kernel_size=kw, strides=2, padding='valid')(c1)
    c1 = KL.LeakyReLU(0.2)(c1)

    c2 = tf.pad(c1, [[0,0],[pad,pad],[pad,pad],[0,0]], mode = 'REFLECT')
    c2 = KL.Conv2D(128, kernel_size=kw, strides=2, padding='valid')(c2)
    c2 = InstanceNormalization(axis=3)(c2)
    c2 = KL.LeakyReLU(0.2)(c2)
```

```
    c3 = tf.pad(c2, [[0,0],[pad,pad],[pad,pad],[0,0]], mode = 'REFLECT')
    c3 = KL.Conv2D(256, kernel_size=kw, strides=2, padding='valid')(c3)
    c3 = InstanceNormalization(axis=3)(c3)
    c3 = KL.LeakyReLU(0.2)(c3)

    c6 = tf.pad(c3, [[0,0],[pad,pad],[pad,pad],[0,0]], mode = 'REFLECT')
    c6 = KL.Conv2D(512, kernel_size=kw, strides=1, padding='valid')(c6)
    c6 = InstanceNormalization(axis=3)(c6)
    c6 = KL.LeakyReLU(0.2)(c6)

    final = tf.pad(c6, [[0,0],[pad,pad],[pad,pad],[0,0]], mode =
'REFLECT')
    final = KL.Conv2D(1, kernel_size=kw, strides=1,
padding='valid')(final)
    model = KM.Model(input_shape,final)
    return model
```

Code snippet 5. 7 PatchGAN Discriminator Implementation

5.4.4. Edge Loss Implementation

As described in section 4.3.4 image must have a sharp edge to judge its clarity. Edge loss measures the edge dissimilarity between generated and ground truth images and this helps the generator to generate an image with a clean edge. Edge loss can be obtained by using Sobel kernel operators as edge extractors along the horizontal and vertical axis. Edge loss can be obtained in two steps. First edge information of synthesized and a real image along the horizontal and vertical axis can be extracted by Sobel kernel operators. After that edge loss between real and synthesized image pairs is computed by the L1 distance between the edge map of the real and restored image as shown in the code snippet 5.8. Then the loss has been utilized to modify network weight.

```
def Edge_loss(fake,real):
    real_edge = tf.image.sobel_edges(real)
    fake_edge = tf.image.sobel_edges(fake)

    real_y = real_edge[0, :, :, :, 0]
    real_x = real_edge[0, :, :, :, 1]

    fake_y = fake_edge[0, :, :, :, 0]
    fake_x = fake_edge[0, :, :, :, 1]

    edge_loss = tf.reduce_sum(tf.abs( real_y - fake_y )) +
       tf.reduce_sum(tf.abs(real_x- fake_x))

    return edge_loss
```

Code snippet 5. 8 Edge Loss Implementation

5.5. Hyperparameter Configuration

Hyperparameters are a configuration such as optimization, epochs, batch size, learning rate, and loss weights that are set before the learning process starts. These configurations are tunable and can straightforwardly influence how the model trains. Optimizations are a method used to change neural network attributes such as weights and learning rate to minimize losses. There are many optimizations such as Adam, stochastic gradient descent, RMSProp, AdaGrad, etc. Adam (Kingma & Ba, 2015) with learning_rate=0.0001,beta_1=0.9, beta_2=0.999 was used in this study since it is designed to combine the advantage of RMSProp and AdaGrad. Additionally, Adam is efficient and requires less memory since it is based on a first-order gradient (Kingma & Ba, 2015). Based on baseline work the model is trained with a batch size of one for 200 hundred epochs. After 150 epochs the learning is linearly decayed to 0.0000077 to stabilize the training. The coefficients γ, α, and β are loss weights for feature, content and edge loss respectively and utilized to adjust the relative importance of the loss functions. Those coefficients have no default value generally they can be tuned between 1 and 100.

Hyperparameter name	Value
Optimization	Adam
Epoch	200
Batch size	1
Learning rate	0.0001
γ	50
α	10
β	12

5.6. Experiment Class

To assess the execution of the proposed model testing the initial work (baseline work) is important. Five different classes of experiments are conducted as shown in table 5.1. The first experiment is the implementation of the baseline work (DeblurGAN-V2) which uses pre-trained InceptionResnet-v2 as a feature extractor. The second experiment (EIDGAN) is the initial work of the proposed model. It replaces InceptionResnet-v2 with EfficientNet-B7 for feature extraction, additionally, edge loss was utilized to preserve edge information. The third experiment (EAIDGAN-Ra) is based on the second experiment. In the third experiment channel and self-,

attention, and the relativistic average loss are introduced to the network as described in the previous section. The fourth experiment (EAIDGAN -LS) is similar to the third experiment the only difference is least square loss is used instead of the relativistic average loss to train the generator and discriminator. Finally, to examine the importance of edge loss EAIDGAN-LS is trained without edge loss (EAIDGAN-LS-WE).

Table 5. 1 List of experimental Classes

Notations	Experiment	Dataset	Model Used
DeblurGAN-V2	Baseline work	GoPro	InceptionResnet-V2
EIDGAN	Initial work of the proposed model without attention mechanism	GoPro	EfficientNet-B7
EAIDGAN-Ra	EIDGAN with attention mechanism and relativistic average discriminator loss	GoPro	EfficientNet-B7
EAIDGAN-LS	EIDGAN with attention mechanism and least square loss	GoPro	EfficientNet-B7
EAIDGAN-LS-WE	EAIDGAN-LS without edge loss	GoPro	EfficientNet-B7

CHAPTER SIX

6. RESULTS AND DISCUSSIONS

6.1. Results And Discussions

Having discussed the details of the implementations and experiments to be performed, this chapter presents the results that were obtained from the experiments. The proposed model performance was tested using the evaluation metrics mentioned in the third chapter. Moreover, the proposed model's performance is compared with existing image deblurring architectures (Deblurgan-v2).

6.2. Image Deblurring

Image deblurring takes a low-level blurry image as an input and generates a sharp image with consideration of preserving feature and edge information. This work conducts a distinctive training experiment to clarify the subjective and quantitative results of comparing the baselines on two different datasets that have been used to train and test the proposed model. These datasets are namely the GoPro (Nah et al., 2017) data set and Lai dataset. After training the model the execution of the model was tested for its ability to reconstruct the original image. To measure the similarity between original and synthesized images the GoPro test set was used, while to conduct human evaluation Lai data set is used since it has not ground truth as described in chapter three.

Five experiments were conducted in this study as described in table 5.1. First, baseline work (DeblurGAN-V2) is implemented. It utilizes InceptionResnet-v2 as the backbone in the generator network, it is based on global and local discriminators. However, as discussed in the result section the score of this experiment was not sufficient and the quality of the image was also not good. To solve this one EIDGAN was implemented by replacing InceptionResnet-v2 with pre-trained EfficientNet-B7 in the generator network and additionally, edge loss was utilized to preserve edge information. In this experiment, the score of the SSIM was improved but the score of the PSNR is smaller than baseline work. To improve the score EAIDGAN-Ra is proposed by adding channel and self-attention to the EIDGAN with a relativistic average loss. EAIDGAN-Ra outperforms the two experiments in terms of qualitative and quantitative measurement. EAIDGAN-LS is the fourth experiment proposed by replacing the relativistic average loss with the least square loss and it outperforms all three experiments. Finally, to examine the importance of edge loss in this work EAIDGAN-LS is trained without edge loss (EAIDGAN-LS-WE), and the model shows the striking drop in performance as appeared in table 6.1 All five experiments were performed in the

same hardware and software, hyperparameters, and dataset configuration as described in section 5.6. All models are trained for 200 epochs with a batch size of 1. The EAIDGAN-LS takes around 3 days to converge.

6.3. Experimental Results

Figure 6.1 shows the result obtained by baseline work (DeblurGan-V2). As shown in the figure the baseline work removes some blur from the blurred image, but still, some details of the image are not visible when compared to the original image. For example, the wood in front of the standing man and the lines around the window is still not close to the original image.

Figure 6. 1 Sample test result by DeblurGAN-V2 model

Figure 6.2 depicts the result obtained by testing EIDGAN. As seen in the figure this model improves the details around the window, however the area around the face of the standing man contains some ringing artifacts, and some lines on window areas were disappeared as shown in cropped image patches.

Figure 6. 2 Sample test result by EIDGAN

Figure 6.3 depicts the results of EAIDGAN-Ra. The figure shows how this experiment improve the quality of the image when compared to the above two experiments. As seen in the figure the blur around the window is improved, and the ringing artifact that occurred on the man's face in the second experiment also improved, but still, the details of the lines in the window area are not visible.

Figure 6. 3 Sample test result by EAIDGAN -Ra

Figure 6.4 explains the results of EAIDGAN-LS. As seen in the figure the quality of an image was improved more than all three experiments. As seen in cropped image patches the lines around the window and the area around the standing man were closer to the original image. In general, this figure demonstrates how the EAIDGAN-LS experiment outperforms the other three works in terms of the quality of the image it generates.

Figure 6. 4 Sample test result by EAIDGAN-LS

6.4. Evaluation Metrics

6.4.1. Objective Evaluation metrics

Two objective evaluation metrics (PSNR and SSIM) were utilized to assess the execution of the model as described in section 3.6. Table 6.1 summarizes the result of the learned model to show their reconstruction ability via PSNR and SSIM metrics to measure the similarity of restored image and ground truth image. Those metrics are calculated by using 1,111 images from the GoPro test sets. The Higher PSNR and SSIM show the restored image became close to the ground truth image.

Table 6. 1 PSNR AND SSIM for all experiments

S.NO	Experiments	PSNR	SSIM	Parameter in million
1	DeblurGAN-V2	25.62125	0.82452	59.6
2	EIDGAN	25.52	0.82588	26.9
3	EAIDGAN-Ra	26.1293	0.8364	33.9
4	EAIDGAN-LS	**26.174**	**0.8406**	33.9
5	EAIDGAN-LS-WE	24.26	0.778	33.9

Bold values indicate the best results in the experiments.

Quantitative Experimental result shows the EAIDGAN-LS outperform other three experiments by PSNR and SSIM score; this indicates the generated image are closer to the real image with shape and feature. In the experiments, EIDGAN sores smaller PSNR than other this is because it has the small number of the parameter when compared to other three models. EAIDGAN-Ra is similar to EIDGAN but an attention mechanism is added to EAIDGAN-Ra and the average PSNR and SSIM are improved. To improve performance more relativistic average loss in EAIDGAN-Ra is replaced by least square loss and forms EAIDGAN-LS. This outperforms all experiments by average PSNR and SSIM. EAIDGAN-LS has the best results giving the highest PSNR and SSIM on the GoPro dataset with a small parameter when compared to baseline work. Finally, to identify the impact of edge loss on this work EAIDGAN-LS trained without edge loss (EAIDGAN-LS-WE) experiment shows a striking drop in performance and this demonstrates how edge loss impacts the performance of this work.

Figure 6. 5 Model comparison on GoPro dataset

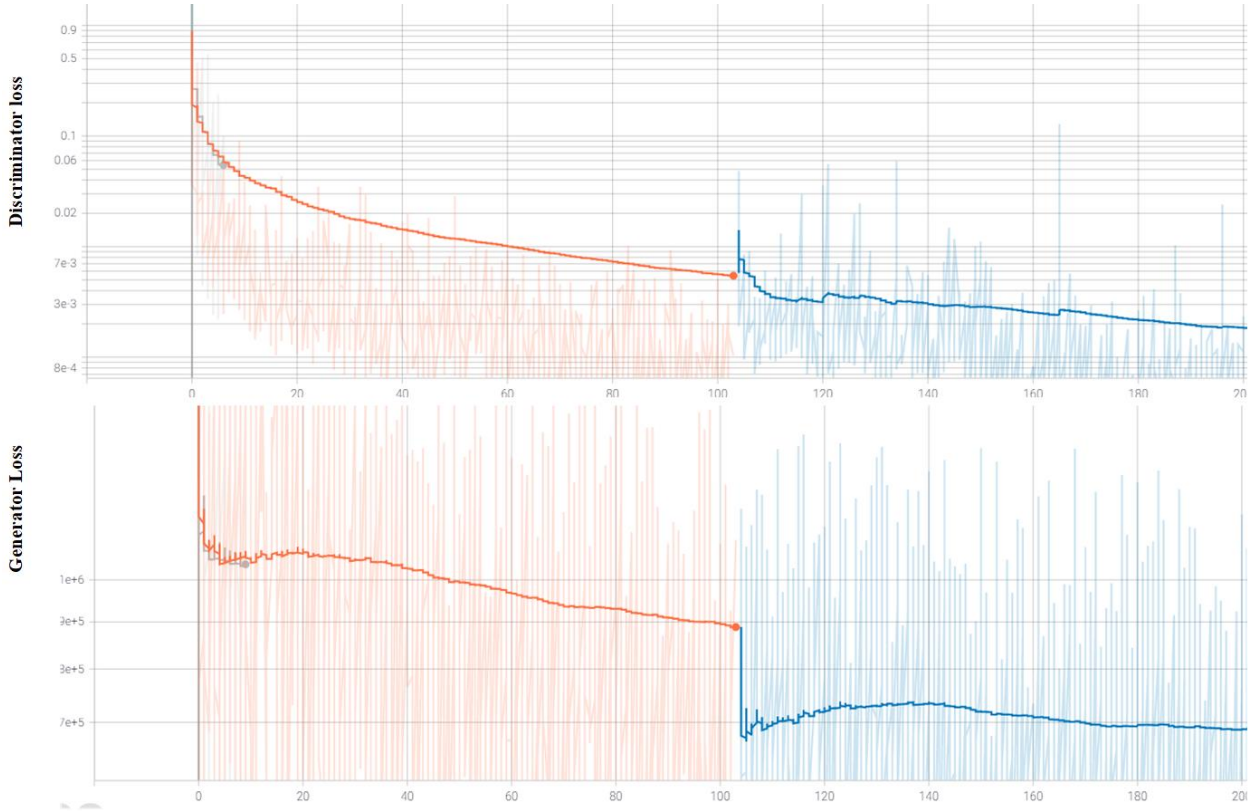


Figure 6. 6 EAIDGAN-LS training loss

6.4.2. Human Evaluation

Since objective evaluation is not sufficient for accessing the quality of an image, so human evaluation is also performed in this work [3]. The interested user fills the questionaries by given two images generated by baseline and proposed model to choose the more natural image from the given two images as depicted in figure 6.7. Higher values indicate the best results.

[3] Human Evaluation User found at:
https://docs.google.com/forms/d/1xzQk2GPzOkiDI6iKRnKsHW08BizIzXL6sYW1LDPztx4/edit

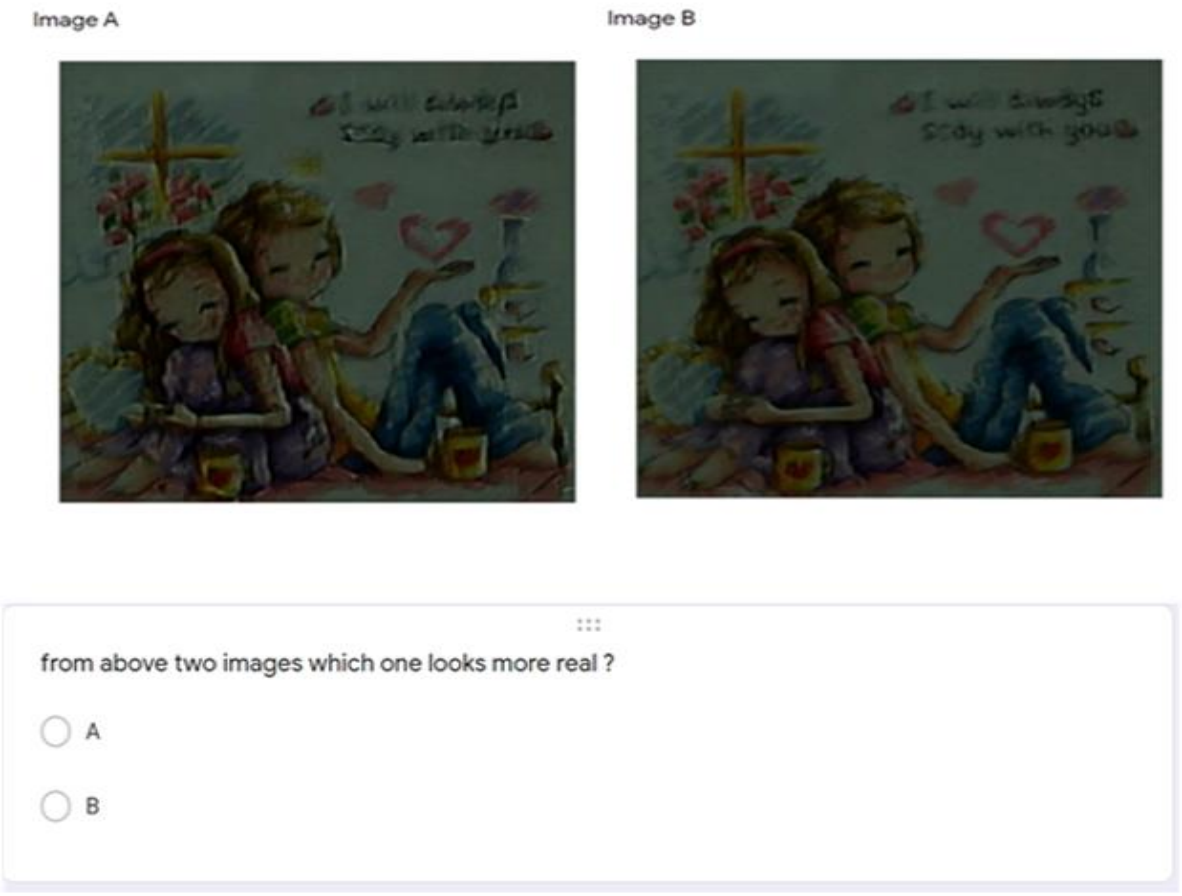


Figure 6. 7 Human evaluation study

Table 6. 2 Summary of a human evaluation

	Q1		**Q2**		**average**	
	A	B	A	B	A	B
User Guess	16	8	14	10	15	9
User Guess/Total User	8/24	16/24	10/24	14/24	9/24=0.375	15/24=0.625

Where A is the DeblurGAN-V2 result and B is the EAIDGAN-LS result. The detailed Human evaluation is discussed in the appendix section of a human evaluation user study.

6.5. Object Detection on YOLO

Object recognition is the most studied topic in computer vision with applications in various fields from autonomous driving to security. In recent years, deep CNN-based approaches have appeared superior execution compared to traditional methods. To solve any computer vision problem with a high-performance CNN algorithm requires a large data image dataset. However, the quality of images was degraded by various artifacts, including motion blur, as described in chapter 1. Before being used by those algorithms a blurry image must be deblurred by the image deblurring algorithm to improve quality. Based on this, to assess the importance of the proposed model in computer vision applications, the confidence and number of an object detected result of the blurry and restored image was evaluated by the YOLO-V3 object detection. Both the blurred and the restored image were sent to the pre-trained YOLO-V3 network and the confidence and the number of objects detected by the blurred image and the restored image were assessed as appeared in Figures 6.8 and 6.9.

Figure 6. 8 Detection result of a blurry image on YOLO-V3

As appeared in figure 6.8 the network only detects two objects with small confidence. This due to the quality of an image is significantly decreased by blur artifact and this indicates how the blur affects the performance of the computer vision algorithm.

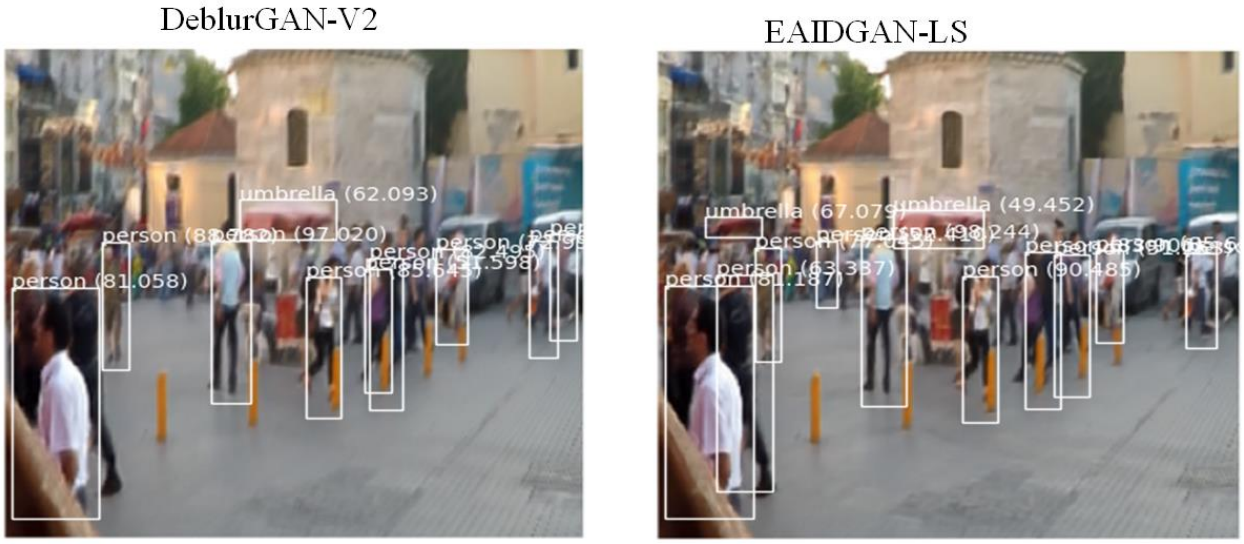


Figure 6. 9 Yolo-V3 detection result of images restored by baseline work and proposed model

Table 6.9 describes the Yolo-V3 detection result of baseline work (left) and the proposed model (right). As appeared in figure 6.9 the number of an object detected and the confidence was significantly improved when compared with the detection result of a blurry image. The number of objects detected in the image restored by baseline work is ten, while the number of objects detected in the image restored by the proposed model is twelve and this indicates that the image deblurred by the proposed method has high quality when compared to the image restored by baseline work.

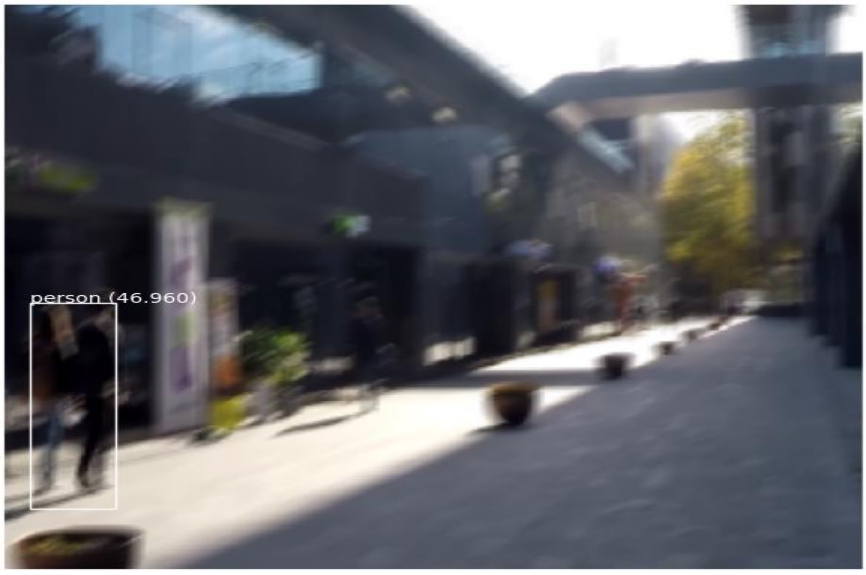


Figure 6. 10 Detection result of a blurry on YOLO-V3

Figure 6.10 shows the Yolo detection result of a blurry image. As appeared in the figure the algorithm only detects one object with small confidence.

Figure 6. 11 Yolo-V3 detection result of images restored by baseline work and proposed model

Figure 6.11 depicts how the image deblurring algorithm improves the detection power of Yolo-V3. The detection result of Yolo-V3 was improved on image deblurred by baseline work (left) and the proposed model (right). As appeared in figure 6.10 Yolo-V3 detects only one object with small confidence while after the image is deblurred Yolo-V3 detects three objects with high confidence when compared to a blurry image. The confidence score of the image restored by the proposed model (right) was higher than the image restored by baseline work (left).

Generally, as the proposed model (EAIDGAN) improves the number of an object detected and the score of confidence when compared to blurry image and image restored by baseline work. This illustrates that the proposed method solves the problem stated in section the statement of the problem to some extent.

6.6. Discussion

6.6.1. Discussion on PSNR Results

For the performance and quality evaluation purpose, a PSNR was performed as appeared in figure 6.7. EIDGAN is implemented without an attention mechanism and the average PSNR of EIDGAN is only 25.52. EAIDGAN-Ra is implemented with attention mechanism and relativistic average

loss and the average PSNR is improved from 25.52 to 26.1293. EAIDGAN-LS is implemented by replacing the relativistic average loss with the least square loss and it beats EAIDGAN-Ra in average PSNR. EAIDGAN-LS beats EAIDGAN-Ra by average PSNR because the least square loss penalizes the samples on the right side of the decision boundary but far from the decision boundary as described in section 2.2.3. As shown in figure 6.12 the PSNR score of EAIDGAN-LS-WE is dropped from 26.174 to 24.26 when EAIDGAN-LS is trained without edge loss. Based on this one, this experiment tells us how the attention mechanism (self and channel), edge, and least square losses are important in this work.

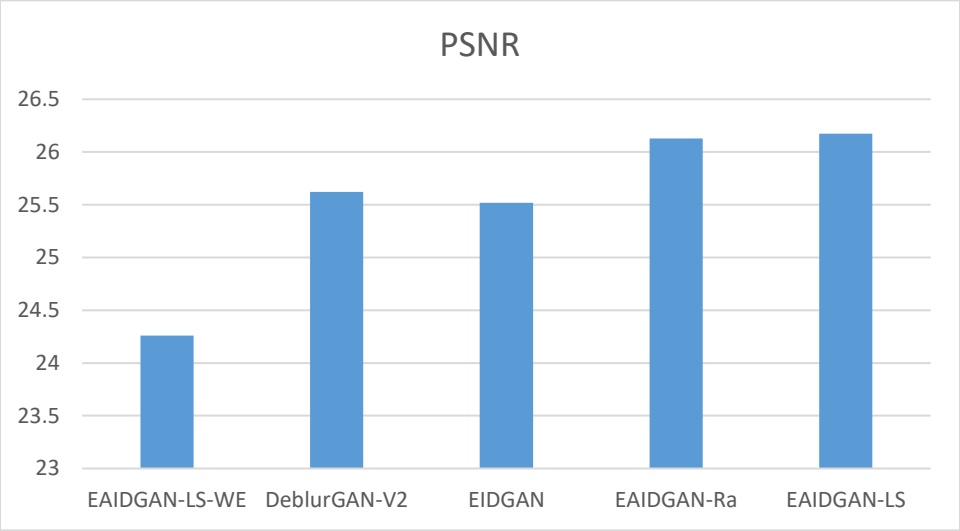


Figure 6. 12 PSNR comparison of the five models

6.6.2. Discussion on SSIM Results

SSIM is another quality evaluation metrics performed on this work to measure structural similarity between real image and restored image based on luminance, brightness, and luminance. The SSIM shows the EAIDGAN-LS beats other models as shown in figure 6.13. This is because of the attention mechanism and least square loss as described in section 6.6.1. As shown in the figure. To access the importance of edge loss EAIDGAN-LS is trained without edge loss (EAIDGAN-LS-WE) and the score is dropped from 0.8406 to 0.778. Based on this observation, it is simple to understand the importance of attention mechanisms (self and channel), and edge loss in image deblurring.

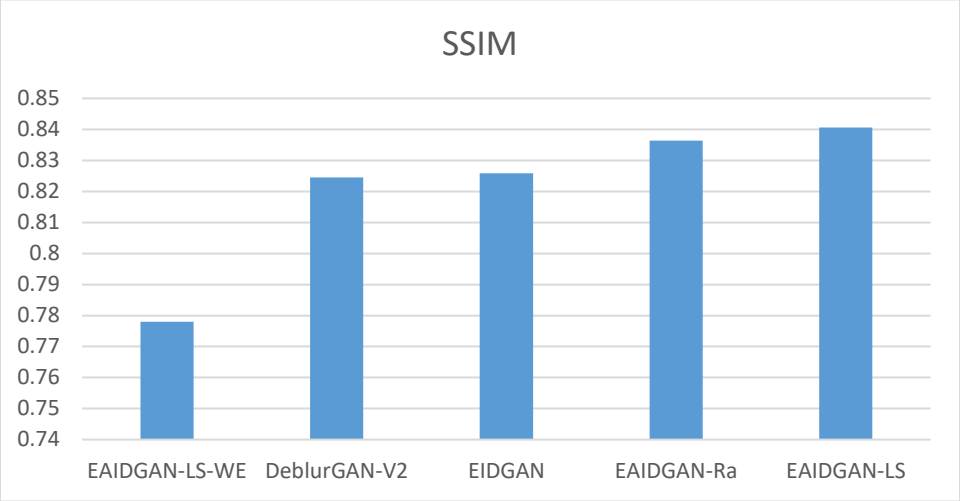


Figure 6. 13 SSIM score of the five models

6.6.3. Discussion on Human Evaluation Results

In this evaluation, users had a chance to compare the more realistic images from baseline and the EAIDGAN-LS model. Based on a human evaluation study some users choose the image restored by DeblurGAN-V2 work as more realistic, while the other users choose the image restored by EAIDGAN-LS as more realistic than baseline work. The average user study chooses the image restored by DeblurGAN-V2 as more real is 37.5%, while the average user chooses the image restored by EAIDGAN-LS as more real is 62.5%. Based on this one, this work tells us the image restored by EAIDGAN-LS is more natural and closer to the real image.

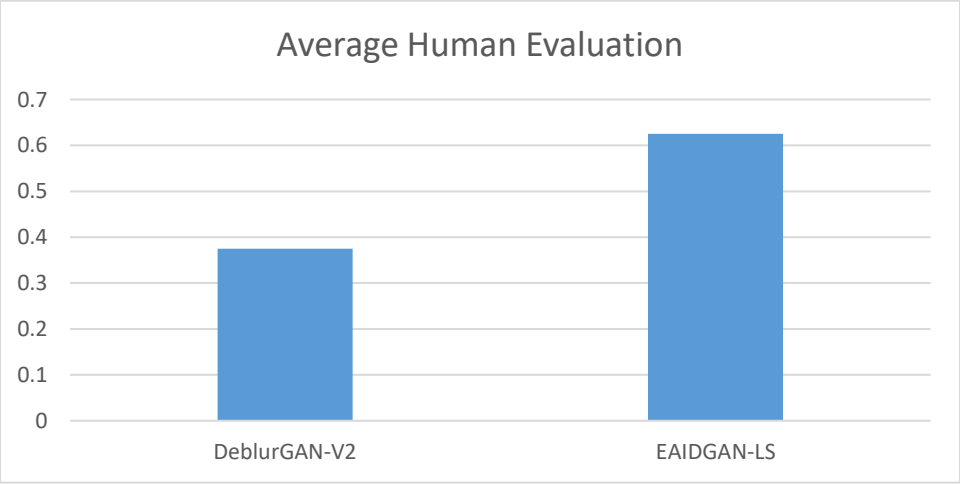


Figure 6. 14 Human evaluation study

CHAPTER SEVEN

7. CONCLUSION AND FUTURE WORKS

7.1. Conclusion

Currently, image-to-image translation has shown successive results in changing an image from one class to another class. This can be applied in various domains such as image deblurring, black and white to the color photo, image super-resolution, edge to photo, and style transfer. Image deblurring is an extension of image-to-image translation that aims to restore sharp images from blurry images. One of the main challenges in image deblurring is it is difficult to handle blur kernels in the image only with convolution kernels since it limits the ability of the network from learning global information. To tackle this problem using an attention mechanism in the deblurring network could be an alternative solution.

This work presented edge and attention-based image deblurring by GAN. The purpose of this study was to improve the quality of the synthesized image by adding a learning function and attention mechanisms (self and channel) to the generator network of baseline work (Kupyn et al., 2019). Among the examination, the goal was to restore the visually realistic image from the blurred image with small computational cost as possible. To do so, this work adds EfficientNet-B7 as a feature extractor to minimize computational cost, channel and self-attention mechanism, and edge loss to the baseline work. Indeed, these modifications make EAIDGAN-LS restore sharp images with clear edges and closer to the original image.

The experiment showed the logical process of the architecture with the mathematical expression utilized to move forward the quality of an image. Out of the five models, the reconstruction result using the EAIDGAN-LS section had better visual quality and performance with different evaluation metrics (PSNR, and SSIM) as shown in the result and appendices section.

EIDGAN is not based on attention mechanism and it has been shown to have poor restoration performance than other models. EAIDGAN-LS has been shown to have better restoration performance than the other four comparison models. As discussed in the result section the score of EAIDGAN-Ra is almost close to EAIDGAN-LS. Both EAIDGAN-Ra and EAIDGAN-LS network is similar and in both networks except the loss function. Both networks utilize attention mechanisms and outperform the other three models and this has answered the first research

question of what is the effect of adding an attention mechanism regarding improving deblurring quality.

The score of EAIDGAN-Ra is almost close to EAIDGAN-LS as shown in the result section. Both networks are similar except EAIDGAN-LS uses the least square loss and EAIDGAN-Ra uses relativistic average loss but, EAIDGAN-LS showed better restoration capability than EAIDGAN-Ra as discussed in the result section. EAIDGAN-LS was trained without edge loss (EAIDGAN-LS-WE) and the model shows the striking drop in performance as appeared in table 6.1. This addresses the second research question of what are the appropriate learning constraints that improve deblurring quality.

Broad subjective and quantitative evaluations illustrate the striking victory of the proposed system against existing methods. The peak signal to noise ratio and structural similarity index measure reaches over 26.174 and 0.8406 respectively which outperforms the other comparative works. Average human assessment study has appeared that this research exceeds expectations at 25% compared to DeblurGAN-v2. Finally, it is reasonable to conclude that adding an attention mechanism (self and channel) and edge loss to the deblurring network would yield a better performance due to they make to makes the network focus on more important futures and exploit the relationship between feature channels and preserve edge information.

7.2. Future Work

As watched within the experiment, the model is emphatically dependent on the attention mechanism (channel and self-attention) and edge loss. Synthesized output depends mainly on attention mechanisms performance as discussed in table 6.1. Based on this one integrating another attention mechanism such as contextual and scale attention to the model can improve the execution of the model. Edge loss is another essential part of the model which has a great impact on the performance of the network as appeared in table 6.1. For computational complexity, the Sobel edge detector was used since it is based on the first-order derivative. So, replacing the Sobel edge detector with another edge detector such as a Laplacian operator can improve the performance of the network. Due to computational resource scarcity, this experiment could not be done on a large-scale dataset. Training the model on a huge dataset would in advance illustrate the effectiveness of the proposed model. Taking the research in this way could make it more robust and fundamental to be utilized on a wide scale.

SPECIAL ACKNOWLEDGMENTS

This research project is funded by Adama Science and Technology University under the grant number:

ASTU/SM-R/238/21

Adama, Ethiopia

REFERENCES

Arjovsky, M., Chintala, S., & Bottou, L. (2017). *Wasserstein GAN*. http://arxiv.org/abs/1701.07875

Avery, K. R., Pan, J., Engler-Pinto, C. C., Wei, Z., Yang, F., Lin, S., Luo, L., & Konson, D. (2014). Fatigue Behavior of Stainless Steel Sheet Specimens at Extremely High Temperatures. *SAE International Journal of Materials and Manufacturing*, *7*(3), 1251–1258. https://doi.org/10.4271/2014-01-0975

Bovik, A., Wang, Z., & Sheikh, H. (2005). *Structural Similarity Based Image Quality Assessment*. *November*, 225–241. https://doi.org/10.1201/9781420027822.ch7

Chakrabarti, A. (2016). A neural approach to blind motion deblurring. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9907 LNCS*, 221–235. https://doi.org/10.1007/978-3-319-46487-9_14

Chaple, G., & Daruwala, R. D. (2014). Design of Sobel operator based image edge detection algorithm on FPGA. *International Conference on Communication and Signal Processing, ICCSP 2014 - Proceedings*, 788–792. https://doi.org/10.1109/ICCSP.2014.6949951

Complete Architectural Details of all EfficientNet Models | by Vardan Agarwal | Towards Data Science. (n.d.). Retrieved August 7, 2021, from https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, *35*(1), 53–65. https://doi.org/10.1109/MSP.2017.2765202

Gonzalez, T. F. (2007). Handbook of approximation algorithms and metaheuristics. *Handbook of Approximation Algorithms and Metaheuristics*, 1–1432. https://doi.org/10.1201/9781420010749

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144. https://doi.org/10.1145/3422622

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Horé, A., & Ziou, D. (2010). Image quality metrics: PSNR vs. SSIM. *Proceedings - International Conference on Pattern Recognition*, *August*, 2366–2369. https://doi.org/10.1109/ICPR.2010.579

Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2020). Squeeze-and-Excitation Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(8), 2011–2023. https://doi.org/10.1109/TPAMI.2019.2913372

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2261–2269.

https://doi.org/10.1109/CVPR.2017.243

Isa, I. S., Sulaiman, S. N., Mustapha, M., & Darus, S. (2015). Evaluating denoising performances of fundamental filters for T2-weighted MRI images. *Procedia Computer Science*, *60*(1), 760–768. https://doi.org/10.1016/j.procs.2015.08.231

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 5967–5976. https://doi.org/10.1109/CVPR.2017.632

Jo, Y., & Park, J. (2019). SC-FEGAN: Face editing generative adversarial network with user's sketch and color. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 1745–1753. https://doi.org/10.1109/ICCV.2019.00183

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9906 LNCS*, 694–711. https://doi.org/10.1007/978-3-319-46475-6_43

Jolicoeur-Martineau, A. (2019). The relativistic discriminator: A key element missing from standard GaN. *7th International Conference on Learning Representations, ICLR 2019*.

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Koh, J., Lee, J., & Yoon, S. (2021). Single-image deblurring with neural networks: A comparative survey. *Computer Vision and Image Understanding*, *203*(December 2019), 103134. https://doi.org/10.1016/j.cviu.2020.103134

Kumar, G., & Bhatia, P. K. (2014). A detailed review of feature extraction in image processing systems. *International Conference on Advanced Computing and Communication Technologies, ACCT*, 5–12. https://doi.org/10.1109/ACCT.2014.74

Kupyn, O., Martyniuk, T., Wu, J., & Wang, Z. (2019). DeblurGAN-v2: Deblurring (orders-of-magnitude) faster and better. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 8877–8886. https://doi.org/10.1109/ICCV.2019.00897

Kwon, S. J. (2011). Artificial neural networks. *Artificial Neural Networks*, *July*, 1–426. https://doi.org/10.15864/jmscm.1104

Lai, W. S., Huang, J. Bin, Ahuja, N., & Yang, M. H. (2017). Deep laplacian pyramid networks for fast and accurate super-resolution. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 5835–5843. https://doi.org/10.1109/CVPR.2017.618

Lai, W. S., Huang, J. Bin, Hu, Z., Ahuja, N., & Yang, M. H. (2016). A Comparative Study for Single Image Blind Deblurring. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 1701–1709. https://doi.org/10.1109/CVPR.2016.188

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

https://doi.org/10.1038/nature14539

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object recognition with gradient-based learning. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1681*(0), 319–345. https://doi.org/10.1007/3-540-46805-6_19

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 105–114. https://doi.org/10.1109/CVPR.2017.19

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 936–944. https://doi.org/10.1109/CVPR.2017.106

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2017). Least Squares Generative Adversarial Networks. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 2813–2821. https://doi.org/10.1109/ICCV.2017.304

Mathew, A., Amudha, P., & Sivakumari, S. (2021). Deep learning techniques: an overview. *Advances in Intelligent Systems and Computing*, *1141*(January), 599–608. https://doi.org/10.1007/978-981-15-3383-9_54

Mirza, M., & Osindero, S. (2014). *Conditional Generative Adversarial Nets*. 1–7. http://arxiv.org/abs/1411.1784

Nah, S., Kim, T. H., & Lee, K. M. (2017). Deep multi-scale convolutional neural network for dynamic scene deblurring. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 257–265. https://doi.org/10.1109/CVPR.2017.35

Ozcinar, C., Rana, A., & Smolic, A. (2019). Super-resolution of Omnidirectional Images Using Adversarial Learning. *IEEE 21st International Workshop on Multimedia Signal Processing, MMSP 2019*. https://doi.org/10.1109/MMSP.2019.8901764

Pan, J., Sun, D., Pfister, H., & Yang, M. H. (2016). Blind image deblurring using dark channel prior. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 1628–1636. https://doi.org/10.1109/CVPR.2016.180

Pang, Y., Lin, J., Qin, T., & Chen, Z. (2021). *Image-to-Image Translation: Methods and Applications*. 1–19. http://arxiv.org/abs/2101.08629

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 1–16.

Rakotonirina, N. C., & Rasoanaivo, A. (2020). ESRGAN+ : Further Improving Enhanced Super-Resolution Generative Adversarial Network. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *2020-May*, 3637–3641.

https://doi.org/10.1109/ICASSP40776.2020.9054071

Review, I. T. A., & Alotaibi, A. (2020). *SS symmetry Deep Generative Adversarial Networks for*.

Rodney, G. H., & López. (2019). Deep Learning with TensorFlow 2 and Keras. In *Packt Publishing* (Issue 2).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-ResNet and the impact of residual connections on learning. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, 4278–4284.

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 10691–10700.

Wang, S. (2017). *Generative Adversarial Networks (GAN): A Gentle Introduction*. *May*, 0–8. http://suwangcompling.com/wp-content/uploads/2017/04/gan_tutorial_suwang.pdf

Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., & Loy, C. C. (2019). ESRGAN: Enhanced super-resolution generative adversarial networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11133 LNCS*, 63–79. https://doi.org/10.1007/978-3-030-11021-5_5

Wu, C., Du, H., Wu, Q., & Zhang, S. (2020). Image text deblurring method based on generative adversarial network. *Electronics (Switzerland)*, *9*(2), 1–14. https://doi.org/10.3390/electronics9020220

Xu, X., Sun, D., Pan, J., Zhang, Y., Pfister, H., & Yang, M. H. (2017). Learning to Super-Resolve Blurry Face and Text Images. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 251–260. https://doi.org/10.1109/ICCV.2017.36

Xu, Z., Wilber, M., Fang, C., Hertzmann, A., & Jin, H. (2018). Learning from Multi-domain Artistic Images for Arbitrary Style Transfer. *ArXiv*.

Yuan, Z., Jiang, M., Wang, Y., Wei, B., Li, Y., Wang, P., Menpes-Smith, W., Niu, Z., & Yang, G. (2020). SARA-GAN: Self-Attention and Relative Average Discriminator Based Generative Adversarial Networks for Fast Compressed Sensing MRI Reconstruction. *Frontiers in Neuroinformatics*, *14*(November). https://doi.org/10.3389/fninf.2020.611666

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019). Self-attention generative adversarial networks. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 12744–12753.

Zhang, S., Cheng, D., Jiang, D., & Kou, Q. (2020). Least squares relativistic generative adversarial network for perceptual super-resolution imaging. *IEEE Access*, *8*, 185198–185208. https://doi.org/10.1109/ACCESS.2020.3030044

Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., & Fu, Y. (2018). Image super-resolution using very deep residual channel attention networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11211 LNCS*, 294–310. https://doi.org/10.1007/978-3-030-01234-2_18

APPENDICES

Appendix A: Sample Dataset

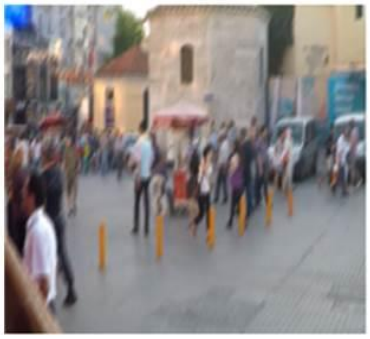
Appendix A.1. GoPro sample Dataset

Appendix A.2. Lai sample Dataset

Appendix B: Result on Different epochs

Appendix B.1: Result of DeblurGAN-V2

Epoch 1



Epoch 10

Epoch 15

Epoch 51

Epoch 101

Epoch 110

Epoch 150

Epoch 170

Epoch 200

Appendix B.2: Result of EIDGAN

Epoch 1

Epoch 10

Epoch 15

Epoch 51

Epoch 101

Epoch 110

Epoch 150

Epoch 170

Epoch 200

Appendix B.3: Result of EAIDGAN-Ra

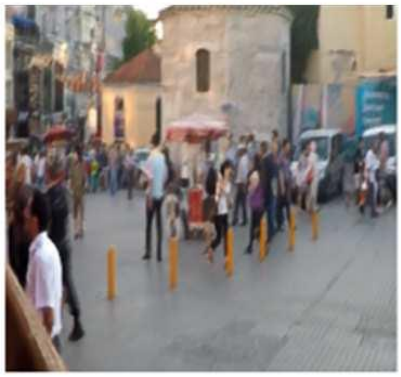
Epoch 1

Epoch 10

Epoch 15

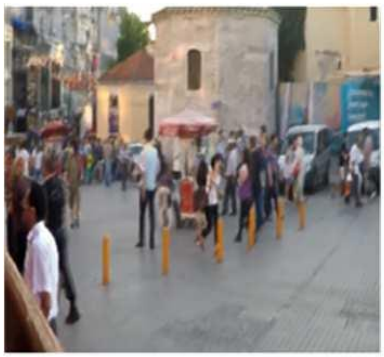
Epoch 51

Epoch 101

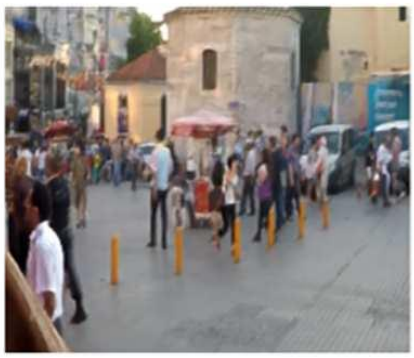


Epoch 110

Epoch 150



Epoch 170



Epoch 200

Appendix B.4: Result of EAIDGAN-LS

Epoch 1

Epoch 10

Epoch 15

Epoch 51

Epoch 101

Epoch 110

Epoch 150

Epoch 170

Epoch 200

Appendix C: Sample Code

Appendix C.1: Dataset preprocessing

```
def select_patch(sharp, blur, patch_size_x, patch_size_y):
    sharp=tf.image.resize(sharp, size=[256,256],method='nearest')
    blur=tf.image.resize(blur, size=[256,256],method='nearest')
    return sharp,blur


def get_dataset_path(dataset_name):
    return Path("datasets") / dataset_name


class IndependantDataLoader:
    def image_dataset(self, images_paths):
        dataset = tf.data.Dataset.from_tensor_slices(images_paths)
        dataset = (
            dataset.map(
                tf.io.read_file,
num_parallel_calls=tf.data.experimental.AUTOTUNE
            )
            .map(tf.image.decode_png,
num_parallel_calls=tf.data.experimental.AUTOTUNE)
            .map(
                lambda x: tf.image.convert_image_dtype(x, tf.float32),
                num_parallel_calls=tf.data.experimental.AUTOTUNE,
            )
            .map(
                lambda x: (x - 0.5) * 2,
                num_parallel_calls=tf.data.experimental.AUTOTUNE,
            )
        )

        return dataset

    def load_test(self,
        dataset_name,
        mode="test",
        batch_size=1,
        patch_size=(256, 256),
        shuffle=False,
    ):
        dataset_path = get_dataset_path(dataset_name) / mode
        print("test data is loaded")
        sharp_images_path = [str(path) for path in
glob.iglob("datasets/gopro/test/B/*.png")]
        blur_images_path = [path.replace("B", "A") for path in
sharp_images_path]
        print(len(sharp_images_path))
        print(len(blur_images_path))

        sharp_dataset = self.image_dataset(sharp_images_path)
        blur_dataset = self.image_dataset(blur_images_path)
```

```
        dataset = tf.data.Dataset.zip((sharp_dataset, blur_dataset))

        if shuffle:
            dataset = dataset.shuffle(buffer_size=200)

        dataset = dataset.map(
            lambda sharp_image, blur_image: select_patch(
                sharp_image, blur_image, patch_size[0], patch_size[1]
            ),
            num_parallel_calls=tf.data.experimental.AUTOTUNE,
        )

        dataset = dataset.batch(batch_size)
        dataset = dataset.repeat()
        dataset =
dataset.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)

        return dataset


    def load(
        self,
        dataset_name,
        mode="train",
        batch_size=1,
        patch_size=(256, 256),
        shuffle=False,
    ):
        dataset_path = get_dataset_path(dataset_name) / mode
        print("train data is loaded")
        sharp_images_path = [str(path) for path in
glob.iglob("datasets/gopro/train/B/*.png")]
        blur_images_path = [path.replace("B", "A") for path in
sharp_images_path]

        sharp_dataset = self.image_dataset(sharp_images_path)
        blur_dataset = self.image_dataset(blur_images_path)


        dataset = tf.data.Dataset.zip((sharp_dataset, blur_dataset))

        if shuffle:
            dataset = dataset.shuffle(buffer_size=200)

        dataset = dataset.map(
            lambda sharp_image, blur_image: select_patch(
                sharp_image, blur_image, patch_size[0], patch_size[1]
            ),
            num_parallel_calls=tf.data.experimental.AUTOTUNE,
        )

        dataset = dataset.batch(batch_size)
        dataset = dataset.repeat()
```

```
        dataset =
dataset.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)

        return dataset
```

Appendix C.2: Train generator and discriminator network

```
@tf.function
def train_step(blurs,sharps,epoch):
  with tf.GradientTape() as gen_tape,tf.GradientTape() as
disc_patch_tape:
    gen_output = generator(blurs, training=True)
    disc_real_output_patch = patch_discriminator(sharps, training=True)
    disc_generated_output_patch = patch_discriminator(gen_output,
training=True)
    gen_total_loss, gan_loss,
perceptual_loss,charbonnier_loss,edge_loss=
generator_loss(disc_generated_output_patch,gen_output, sharps)


    disc_loss_patch = discriminator_loss(disc_real_output_patch,
disc_generated_output_patch)
    disk_t = disc_loss_patch
  generator_gradients = gen_tape.gradient(gen_total_loss,
                                          generator.trainable_variables)
  discriminator_gradients_patch =
disc_patch_tape.gradient(disc_loss_patch,

patch_discriminator.trainable_variables)
  generator_optimizer.apply_gradients(zip(generator_gradients,

generator.trainable_variables))
patch_discriminator_optimizer.apply_gradients(zip(discriminator_gradient
s_patch, patch_discriminator.trainable_variables))


  with summary_writer.as_default():
    tf.summary.scalar('gen_total_loss', gen_total_loss, step=epoch)
    tf.summary.scalar('disk_t', disk_t, step=epoch)
def train(epochs,train_dataset,test_dataset):
```

```
  current_learning_rate = LR
  print(current_learning_rate)
  for epoch in range(epochs):
    start = time.time()
    #reduce LR by at every epochs
    if epoch >= 150:
        current_learning_rate =
learning_rate_decay(current_learning_rate)
        print('current_learning_rate %f' % (current_learning_rate))
        set_learning_rate(current_learning_rate)
    for sharps, blurs in test_dataset.take(1):
        generate_images(generator,blurs,sharps,epoch)
    # Save the model every 15 epochs

    for index, (sharp_images, blur_images) in enumerate(train_dataset):
        if index > 2000:
          break
        train_step(blur_images,sharp_images,epoch)
    # Produce images for the GIF as you go
    if (epoch + 1) % 1 == 0:
      save_model(generator,patch_discriminator,epoch)
    print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-
start))
#save the model at the end of each epochs
def save_model(generator,patch_discriminator,epoch):


  generator.save("drakmo/generator_{:04d}.h5".format(epoch))
  #discriminator.save("drakmo/discriminator{:04d}.h5".format(epoch))

patch_discriminator.save("drakmo/patch_discriminator{:04d}.h5".format(ep
och))
#load train dataset
train_dataset = IndependantDataLoader().load(
        "gopro", patch_size=(256, 256), batch_size=1, mode="train")
```

```
test_dataset=IndependantDataLoader().load_test(
        "gopro", patch_size=(256, 256), batch_size=1, mode="test"
    )
train(200,train_dataset,test_dataset)
```

Appendix C.3: Edge Loss

```
def Edge_loss(fake,real):
    real_edge = tf.image.sobel_edges(real)
    fake_edge = tf.image.sobel_edges(fake)
    real_y = real_edge[0, :, :, :, 0]
    real_x = real_edge[0, :, :, :, 1]
    fake_y = fake_edge[0, :, :, :, 0]
    fake_x = fake_edge[0, :, :, :, 1]
    edge_loss = tf.reduce_sum(tf.abs( real_y - fake_y )) +
tf.reduce_sum(tf.abs(real_x- fake_x))
    return edge_loss
edge_loss = Edge_loss(gen_output, target)
```

Appendix E: Human Evaluation User Study

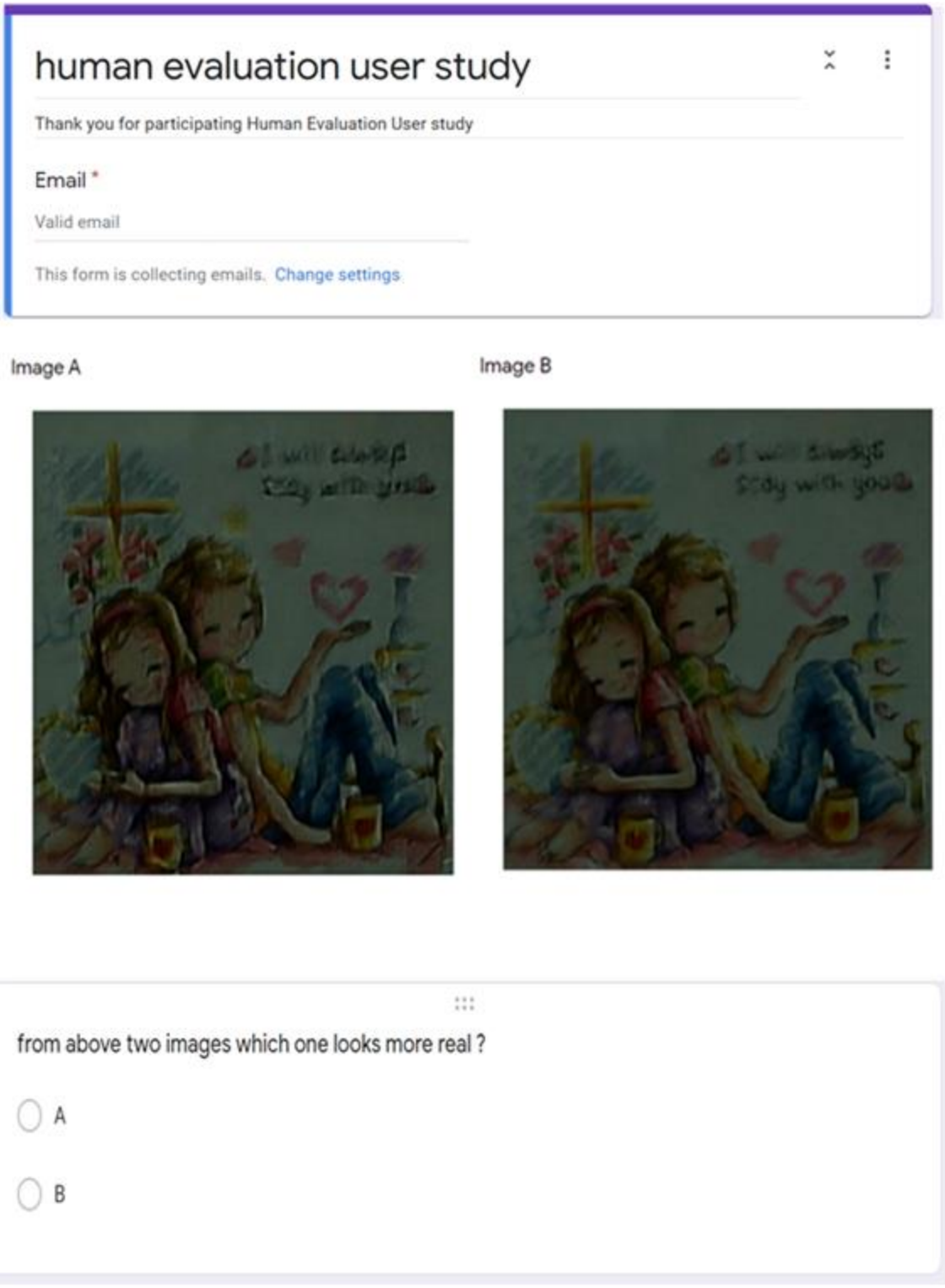


Image A

Image B

from above two images which one looks more real ?

- ○ A
- ○ B

S.NO	Name of Evaluator		Q1	Q2
1	Merga		B	B
2	Muktar		B	B
3	Yordanos		A	B
4	Dagmawit		B	B
5	Mengistu		A	A
6	Dedefo		B	B
7	Seid		B	A
8	Kidist		A	A
9	Biruk		B	B
10	Chala		B	B
11	Diriba		B	B
12	Bahradin		B	A
13	Fayyisa		B	B
14	Bikila		B	B
15	Anteneh		A	B
16	Ebisa		B	A
17	Biniyam		A	A
18	Yadeta		B	B
19	abera		A	A
20	Dagne		B	B
21	Felmi		A	A
22	Lata		B	B
23	Deyyas		B	A
24	Tihtina		A	A
Total		A	8/24 = 0.333	10/24 = 0.416
		B	16/24 = 0.66	14/24 = 0.583

Where A is the result of DeblurGAN-v2 and B is the result of EAIDGAN -LS