Learned Image Compression in Low Bitrate based on Relevant Properties prioritized using Generative Adversarial Network

Ziyad Ahmed Abdulla

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 2023

Adama, Ethiopia

Learned Image Compression in Low Bitrate based on Relevant Properties prioritized using Generative Adversarial Network

Ziyad Ahmed Abdulla

Advisor: Worku Jifara (PhD)

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 2023

Adama, Ethiopia

Declaration

I hereby declare that this Master Thesis entitled "**Learned Image Compression in Low Bitrate based on Relevant Properties prioritized using 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.

Ziyad Ahmed Abdulla	_____	_____
Name of student	Signature	Date

Recommendation of Advisors

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Learned Image Compression in Low Bitrate based on Relevant Properties prioritized using Generative Adversarial Network"** prepared under my guidance by **Ziyad Ahmed Abdulla** submitted in partial fulfillment of the requirements for the degree of Masters 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.

Worku Jifara (PhD)	_____	_____
Major Advisor	Signature	Date

Approval Page

I, the advisor of the thesis entitled “Learned Image Compression based Relevant on Properties Prioritized using Generative Adversarial Network” developed by Ziyad Ahmed Abdulla, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Worku Jifara (PhD) _____ _____

Major Advisor Signature Date

We, the undersigned, members of the Board of Examiners of the thesis by **Ziyad Ahmed Abdulla** have read and evaluated the thesis entitled “**Learned Image Compression in Low Bitrate based on Relevant Properties prioritized using Generative Adversarial Network”** and examined the candidate during 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 Computer Science and Engineering.

_____ _____ _____

Chairperson 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).

_____ _____ _____

Department Head Signature Date

_____ _____ _____

School Dean Signature Date

_____ _____ _____

Office of Postgraduate Studies, Dean Signature Date

Acknowledgements

In first and foremost, I wish to convey my deepest appreciation to the Almighty Allah for giving upon me the strength, wisdom, determination, and unwavering perseverance necessary to embark on this research effort.

My heartfelt gratitude goes to my advisor, Dr. Worku Jifara, for his invaluable guidance, unwavering support, and mentorship throughout the entirety of this study. His expertise and insightful feedback have been pivotal in the successful culmination of this research.

I extend my profound thanks to my family, whose boundless love, encouragement, and unwavering belief in my capabilities have provided an indispensable foundation for my academic journey. Their steadfast support and understanding have been essential pillars of my educational pursuit. I also wish to express my appreciation to my friends, whose encouragement, motivation, and assistance have accompanied me throughout the research journey, serving as a source of inspiration and enhancing the overall experience.

Finally, my gratitude extends to all individuals, institutions, and organizations who have contributed, directly or indirectly, to the completion of this research. Your support has been a cornerstone of the successful realization of this study. I'd like to acknowledge both those who provided positive encouragement and those who offered constructive criticism, as each has played a role in my journey.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AE	Auto Encoders
GAN	Generative Adversarial Network
VAE	Variation Auto Encoder
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
RAM	Random Access Memory
RNN	Recurrent Neural Network
JPEG	Joint Photographic Experts Group
MSE	Mean Squared Error
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index Measure
EMP	Earth Mover Distance
WGAN	Wasserstein Generative Adversarial Network
GC	Global Compression
SC	Selective generative compression
ReLU	Rectified Linear Unit
BPG	Better Portable Graphics
DCT	Discrete Cosine Transform
DWT	Discrete Wavelets Transform

Abstract

The primary idea behind lossy image compression is to reduce image size. Image information is separated into three categories in image compression techniques: irrelevant, relevant, and redundant image data. Lossy image compression removes irrelevant and redundant information, leaving only relevant information. Existing image compression methods face challenges in maintaining visual quality at low bitrates below 0.1 bpp. Recent learned compression methods have shown promise in extremely low bitrate compression. However, most techniques treat the image holistically without considering the relevance of different visual properties. In this research, we developed an efficient learned compression approach using GAN architecture that prioritizes the compression of relevant features containing important visual information. Two important features - color and grayscale - were extracted from images and assigned varying priorities for compression. We choose two images features based on their importance in different areas to change lossy images, reduce image size, and generate image quality based on user needs. The proposed method utilized two networks in the generating module: compressor and decompressor. The compressor network takes a feature image and compresses it based on the priority set. The decompressed features were then combined to reconstruct the images. Experiments conducted on COCCO dataset showed the proposed technique with color prioritization achieved 30.5 PSNR and 0.84 SSIM at 0.02 bpp, significantly improving over baseline. The proposed Learned Image Compression in Low Bitrate based on Selecting important Feature to give Priority using Generative Adversarial Network was compared to GAN for Extreme Learned Image Compression and GAN for Fidelity-Controllable Extreme Image Compression. Our work results demonstrate the efficacy of the proposed technique in improving visual quality and compression ratio by selective feature prioritization, while avoiding commonly faced problems like artifacts at very low bitrates. Selective prioritization of relevant features like color, and grayscale enables efficient low bitrate image compression while improving visual quality. The approach can facilitate storage and transmission of images in bandwidth-limited applications.

Keywords: *Learned Image compression, Generative adversarial network, compression ratio, compressor, Decompressor, priorities, sematic level.*

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

In this modern era of big data (Jamil, Piran, and MuhibUrRahman 2022), the size of the images data was the biggest concern for image processing and researchers' areas. Image data was one of the contents of big data. The quality of the camera (Glover-Kapfer et al. 2019), (Preti, Verheggen, and Angeli 2021) had drastically increased to capture high-quality images this quality of image increased the size. The quality of image data was important to computers to make decisions in different field's area to help humans. This field areas were computer robotics, e-commerce, education area, medical area, communication network and, essential uses image data. However, the quality images very important to make decision, (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019; Jamil et al. 2022; Lu et al. n.d.; Torfason et al. 2018a; Wu, Huang, and Shen 2020; Zeegers et al. 2020) this quality image was occupied large memory space and consumed a large bandwidth for transmission time over the network.

Image compression was,(Hu et al. 2022) a process applied to a visuals file to minimize its size in bytes without lessen an acceptable threshold image quality. The basic principle for image compression was to reduction of size images. Image compression (Jamil et al. 2022) was one of the most useful and commercially successful technologies in the field of digital image processing. Image compression have two types of compression techniques (Jamil et al. 2022)(Hussain, Al-Fayadh, and Radi 2018)were Lossy and Lossless. Lossless image compression is a technique used for compression of an image where no information is lost. Lossless image compression techniques (Hussain et al. 2018) are very efficient for small-size data. Lossy image compression means on the other hand, minimize the size of images by removing irrelevant information and reducing redundant information, some data from the original image which could never be recovered. In lossy compression, it is impossible to restore original information removal from the image.

The concept of lossy compression started when (Wallace 1991) proposed the JPEG standard for image compression. Lossy compression involved the minimize of an image size usually by removing small details that required a large amount of data to store at full fidelity, the image compression done in different algorithms (Jamil et al. 2022), including machine learning and deep learning algorithms. The deep learning algorithms such as Auto encoder, variation auto encoder

(VAE), CNN, RNN and GAN were already making significant progress in image compression. More deep learning algorithms were focused on improved to minimize the size of image and generate quality images.

Many researchers (Jamil et al. 2022) (Skodras et al. 2001), (Akutsu et al. 2020; Dua et al. 2021; Wu et al. 2020) can try to do lossy image compression by using different deep learning algorithms. In the above paper, we focus on the very low bitrate below 0.1 bit per pixel (bpp) compression. The authors (Tschannen et al. 2018) proposed extreme low image compression less than 0.1, which can produce image quality used realistic and artifact-free reconstructions at very low bitrates by learning to synthesize image content. The paper (Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019) shows that at low bitrates image compressed, in this paper produces visually quality results were completely generates parts of the image from a semantic label map. The work (Akutsu et al. 2020) proposed block base image compression, uses a selective detail decoder that is trained by a weighted MS-SSIM loss. The selective detail decoder is specialized for specific important parts of the image and can decode them more clearly. These outer (Zhang, Jiao, and Wu n.d.) Used CNN an algorithm with attention-guided dual layer image compression. In this work define CNN module to predict pixels and near a saliency sketch of region of Interest (ROI) that were perceptual quality of image. Another researcher proposed (Akutsu et al. 2020) GAN based image compression in ultra-low bitrate image compression, as relevant information from the image select text and phase comprise and decompress perceptual quality image.

Based on the above study, researcher focus generating high quality image from low bitrate, uses sematic label to gate quality of image and uses more than two or three decoder and encoder. In this algorithm it happens some problem complexity of model, increase the size of compressed image. Another algorithm focused to achieves a highly compressed image low bit rate. In this algorithm essential information needed for reconstruction quality image is removed during removing irrelevant information. Our works focus on loss image compression to give priority for loss important information during compression of an image.

We proposed GAN network architecture for image compression by selecting important feature of image that contain more information of the image and give priority depending on the area that image used. The goal of our research is to develop lossy image compression model to minimize image size in low bitrate and generate quality image. Our work highly compresses the features that

are given a low priority and slightly compresses the features that are given a high priority. According in recently GAN algorithms used for loss image compression on very low bitrate and generate quality of images from low bitrate (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019; Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019; Iwai et al. 2020; Quan, Nguyen-Duc, and Jeong 2018). Our work highly compresses the features that are given a low priority and slightly compresses the features that are given a high priority. Additionally, they compared our work results with deep learning loss image compression in low bitrate methods and state-of-the-art deep learning-based loss image compression methods.

1.2. Motivation of the Study

The main motive of this study comes from the need to improve efficiency of loss image compression method. The primary objective was (Agustsson et al. n.d.; Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019) to improve minimize the size of images and generate the quality of images that contains important feature. To achieve this goal deep learning help to compress image size less than 0.1 and generate the quality of images. It was also help to the user to solve the storage problem and transition time through a network. Although there (Agustsson et al. n.d.; Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019; Iwai et al. 2020)has been development in image compression at low bitrates of less than 0.1 that has produced images of excellent quality, it is surprising that there is still potential for improvement depending on consumer need. In order to bridge the gap between the quality of images captured by high-quality devices and deep learning's ability to use its potential to minimize the size of images and generate them again, we are motivated to conduct research on lossy image compression in low bitrate.

Generally, our motivation for researching on Image compression in low bitrate started from proposes to make storage usage effective by compress image size into low bitrates, improve image quality by generating quality of images. And also enable to improve compression ratio, generated quality of images and transmission through a network. Finally, our goal is to develop image compression model that contribute to the development of more effective compression image into low bitrate and generate quality images.

1.3. Statement of the Problem

Learned image compression has been ongoing for many years, loss images received much attention in research areas for many years, and many well-known compression standards have been created. However, the size and quality of images generated image in learned lossy image compression in low bitrate was a challenge. To overcome this challenges extremely low bitrate image compression was done in terms of compression ratio and generating quality of images(Agustsson, Tschannen, Mentzer, Timofte, et al. 2019). However, in this works not consider to important feature of images used to reconstruct quality of compressed images properties of images, then affect the performance of reconstructed images. Extremely low bitrate image compression focuses on goal of minimizing image compression size and to generate quality images used somatic label of the original image. In (Iwai et al. 2020)this lossy image compression works based context of the images, visual quality performed poorly when compression ratio was high. During the minimize size of compressed image, important information used for image reconstruction was lost. This increased noise and blur while decreasing visual quality of reconstructed image.

The main problem addressed in this paper is in the inadequate the performance of the existing compression in low bitrate and generating quality image techniques. To solve compression problems by giving priority to removing irrelevant properties, they used less important properties to represent lower bits and more important properties to represent more bits. Even though considering the background information as irrelevant or less irrelevant information was a good idea, other properties should also get attention during image compression model development based on their relevancy in each area or location before storing or transmitting over network.

1.4. Research Questions

In this study research, we shall try to answer to the following questions.

RQ1: Which relevant properties should be selected and prioritized as important features in low-bitrate learned image compression using Generative Adversarial Networks (GANs)?

RQ2: How does the prioritization of important features affect image compression and the resulting image quality in low-bitrate when implemented with (GANs)?

1.5. Objective of the Study

1.5.1. General Objective

The main objective of the study was to develop a model for Prioritization of Relevant Properties for Improvement of Low-Bitrate Learned Image Compression Using Generative Adversarial Networks.

1.5.2. Specific Objectives

The following specific objectives will be addressed to achieve the general objective.

- To extract important feature of image and give priority as relevance.
- To develop the proposed architecture by improving generator model compressor and decompressor networks according to given priority of features for image compression.
- To train the model by given dataset according to give priority feature of images to enhance compression ratio and generate quality of images.
- To evaluate the performance of the model based on test dataset used SSIM and PSNR metrics.

1.6. Significance of the Study

Lossy image compression plays a critical role where there is a need for images to be stored, transmitted, by minimizing the size of images and reconstruct to quality images. Our works low bitrate image compression uses of important feature of images as relevant properties to give priority. It can be applied to a wide range of application domains by lowering image sizes, improving data transport efficiency. To generating quality of images according to the user need to give priority to the feature was generate the quality of images and applies in deferent area according to the given priorities. The other area of image data can be stored on your computer, social media and dataset server of big image data stored on used.

1.7. Scope and Limitations

1.7.1. Scope of the Study

This thesis focuses on developing a model of a generative adversarial network for efficient image compression in a low bit rate of less than 0.1 and generating the quality of images. This work is focuses on developing an architecture for efficient image compression in very low bitrate and

reconstruct almost similar to input image based on properties of image. Our works select features according to the importance of images in different areas we select gray and color features to compress and give priority to these features accordingly.

1.7.2. Limitation of the Study

This thesis work didn't consider lossless image compression and loss image compression, in compression ratio bit per pixel greeter than 0.1. In our works selectin feature of images according to the importance to select only gray and color, due to time and resource limitations.

1.8. Contribution of the Study

Our contribution involved implementing important feature as relevant properties to prioritize image compression in the generator and discriminator of the GAN in the name of RPP-GAN.

- We applied priority of the feature in compressor network to compress the feature according to given priorities in the generator's architecture this works greatly minimize the size of images used to transitions time and memory storage for real-world applications.
- We implemented a decompressor approach in the generator module to decompress parallel the feature compressed or generate all feature accordingly.
- We employee the custom layer multiplication to multiply generated feature to produce generated image.

1.9. Organization of this Report

This report has seven chapters. The first one gives an overview of the report and its objectives. The second one reviews the literature on image compression, both traditional and deep learning-based, as well as image-to-image translation and generative adversarial networks. The third one explains the research methodology, including the data collection and evaluation methods. The fourth one presents the proposed model in detail, including its architecture, loss functions, and training process. The fifth one shows the implementation of the proposed model and experimental class. The sixth one analyzes and compares the results of the proposed model with other works. The seventh one concludes the report and suggests future directions of the study.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORKS

2.1. Overview of Compression

This chapter reviews literature and works that are related to our research. We begin with a general overview of digital images and image compression, including types of image compression and techniques used for image compression. Categories image compression techniques: conventional algorithm, deep learning algorithms, like CNN, AE, VAE, and generative adversarial networks (GAN). Finally, we will summarize related works.

2.1.1. Image Compression

An image describes how something appears visually(Virtanen, Nuutinen, and Häkkinen 2022) it provides information about the color, texture, shape, and more at every point in the image. These components make up the visual image. Certain(Virtanen et al. 2022) aspects of an image, such as texture, line, shape, form, color, scale, and depth, are important for a variety of reasons. The amount of color and grayscale in an image must frequently be considered while making judgments in a variety of businesses. The placement of pixels in an image also conveys details about the object's structure. There is additionally metadata, or additional data about an image.

Depending on the application area of image, the intended purpose may choose between grayscale, color, texture and shapes features, as combination of color with other features approach has advantages and disadvantages. For instance, while need more data and processing authority, (Patil et al. 2013)whereas color and gray scale to combine to produce quality of images. Where color with texture or with other produce higher minimize the size of images or may lose some image details. The particular context and limitations of the picture compression task have an impact on the best appropriate solution choice in image compression.

Image compression was an essential part of image-to-image translation techniques (Hu et al. 2022) that minimized the size of images by eliminating unnecessary information and deleting redundant data while reconstructing the images with quality that was as close as possible to the original and precise image. These were accomplished using a variety of methods designed to reduce redundant and unimportant information in the image, resulting in a smaller file size. Compression is widely utilized in many applications, including image databases, networked systems, and mobile devices

because it must be used to reduce both transmission and storage needs. Lossy compression, one of the most often used methods for compressing images, entails throwing away some of the image data to shrink the size of the image. The JPEG (Joint Photographic Experts Group) standard (Skodras et al. 2001), which uses a discrete cosine transform (DCT) to compress the image data, was the most widely used lossy compression method.

The JPEG standard, however, was not without its flaws, and research into additional image compression methods that could deliver improved performance was still ongoing. For many years, academics (T. Chen et al. 2021) had been concerned about still images the compression. At first, picture compression used traditional algorithms. These common algorithms had drawbacks. The constraint of conventional algorithms to produce high-quality images and reduce image size at a very low bitrate was addressed by the deep learning method.

2.1.2. Types of Compression

There are two main types of image compression (Hussain et al. 2018), lossless and lossy compression algorithms. Lossless compression algorithms reduce the file size of an image without losing any information, resulting in a compressed image identical to the original. This is achieved by encoding the data in a more efficient way, such as using variable-length coding or predictive coding. Common lossless compression algorithms [(Hussain et al. 2018), include GIF and PNG.

Lossy compression algorithms (Hu et al. 2022),(Hussain et al. 2018), on the other hand, achieve higher compression ratios by discarding some of the image's information deemed less important or less visible to the human eye. This can result in a perceptible loss of quality or detail in the image but can also significantly reduce the file size. Common lossy compression algorithms include JPEG and MPEG. The compression algorithm depends on the specific requirements of the application, such as the desired compression ratio, visual quality, and computational complexity.

Lossy image compression has achieved significantly (Hu et al. 2022; Tschannen et al. 2018; Wu et al. 2020) a higher compression ratio than lossless image compression. Because lossy image compression algorithms could remove some of the important information of the image, which was deemed less important or less visible to the human eye that reduced image size. In addition, lossy image compression was more computationally efficient than lossless image compression. Lossy image compression algorithms require less computational time to encode and decode the image. However, the other advantage of the lossy image compression algorithm introduces perceptual

lossy of image quality in detail of the image which would be desirable in some application area. Therefore, the choice of compression algorithm depends on the specific requirements of the application, such as the desired compression ratio, and visual quality of image.

2.2. Image compression techniques

2.2.1. Traditional image Compression Algorithms

Huffman coding (Hu et al. 2022), and arithmetic coding are two early methods for image compression that primarily achieve compression by directly utilizing entropy coding to remove statistical redundancy within the image. Prediction and quantization strategies are described to eliminate spatial and visual redundancy in images, in addition to statistical redundancy using entropy coding and transform techniques. Due to the incorporation of its past coding efforts, JPEG—the most widely used image compression standard—was first suggested (Wallace 1991), as an effective picture compression system. The image is divided into blocks, each of which is then translated into the DCT domain.

To remove visual redundancy, a specific quantization table is utilized, which is designed to maintain low-frequency information while removing more high-frequency (noise-like) details since humans are less sensitive to information loss in high-frequency regions. JPEG 2000 (Skodras et al. 2001), another well-known still image compression standard, represents images in a compact manner using the 2D wavelet transform rather than DCT, and applies an efficient arithmetic coding approach, to minimize statistical redundancy in wavelet coefficients. The basic framework of JPEG compression and decompression is shown in Figure 2.1. JPG uses Static algorithms (Skodras et al. 2001)(Ballé, Laparra, and Simoncelli 2016), also known as fixed algorithms, are a class of image compression techniques that apply a fixed compression algorithm to all images, regardless of their content or characteristics. These algorithms typically operate on a block or pixel basis and use predefined rules to transform the pixel values into a more compact representation.

JPEG was a prevalent choice for compressing still images until the introduction of a more advanced iteration called JPEG-2000 (Skodras et al. 2001). JPEG-2000 employs the Discrete Wavelet Transform (DWT) in lieu of the Discrete Cosine Transform (DCT). Figure 2.2 depicts the foundational structure of JPEG-2000, which aimed to enhance network architecture but didn't fully resolve the issue of visual quality within this context. Nevertheless, traditional algorithms

continue to exhibit certain constraints. Machine learning has stepped in to address various challenges, including the enhancement of visual quality in image compression and the reduction of compression ratios.

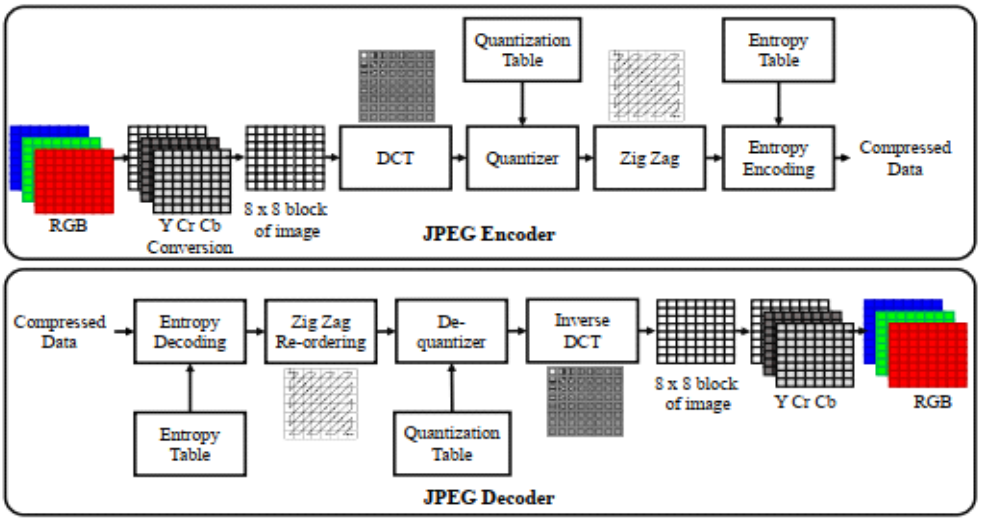

Figure 2. 1 jpg Architecture ((Wallace 1991)

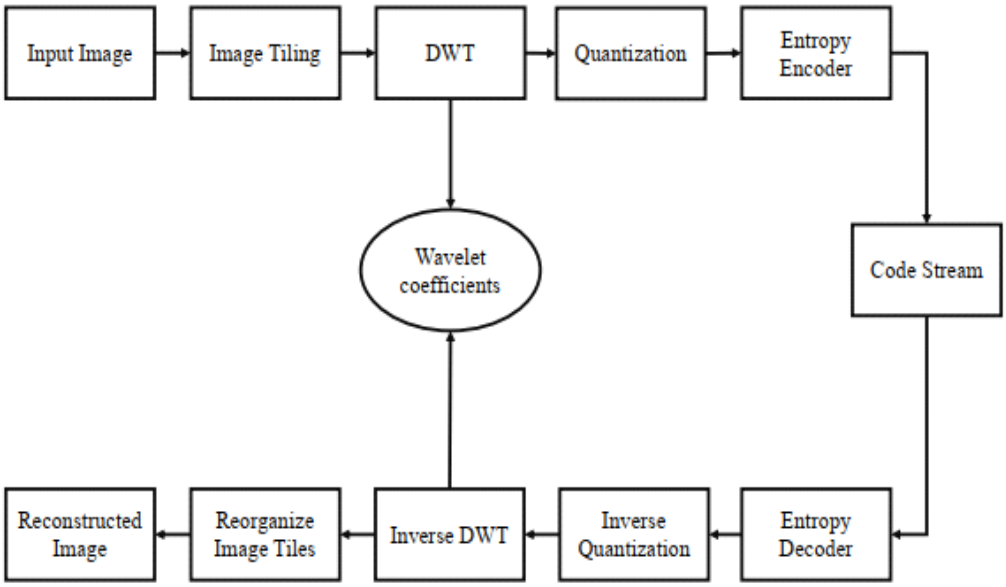

Figure 2. 2 JPEG-2000 compression and decompression Architecture

2.2.2. Deep learning algorithms

Recently deep learning was the emergence of deep image compression as an active research field. Auto-encoders, convolutional neural networks, Variational Auto encoders, recurrent neural networks (RNNs), and generative adversarial networks (GAN) (Jamil et al. 2022; Tschannen et al. 2018) are now the most widely used DNN architectures for this task. The input image is converted by these DNNs into a bit-stream, which is (Hussain et al. 2018) then lossless compressed using entropy coding techniques like Huffman coding or arithmetic coding. Many deep compression techniques rely on context models to record the distribution of the bit stream to lower coding rates.

The learning-driven lossy image compression model that was most frequently utilized by researchers was the auto encoder. The authors of (Sento 2017), compressed the modified National Institute of Standards and Technology (MNIST) images using AE with a Kalman filter. For images in the RGB color space, the model was unable to perform better. They used AE to compress images with a high resolution, but the compression ratio and reduced size were not performed. The non-recurrent design of three layers in combination with the Kalman filter could only be used for grayscale images and did not treat lost data as useless.

Due to their strengths in feature extraction, CNN deep learning models are crucial in image processing. Additionally, CNN is used to compress images. These researchers used CNN to compress images (Cheng et al. 2020)(T. Chen et al. 2021)(Li and Liu 2019), The authors of (Li and Liu 2019), compressed images using CNNs (Convolutional Neural Network). In terms of peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM), these CNN-based image compression models fared better than JPEG and JPEG-2000. A novel idea for a multi-spectral transform for multi-spectral image compression using CNN was presented in (Li and Liu 2019), it was more efficient at compressing data than cutting-edge anchors like JPEG-2000. Likewise, (Dua et al. 2021), they proposed a data reduction CNN for the compression of hyperspectral images but had limitations on the reconstruction quality of images at low bitrate.

VAE is a variation of NN that is related to probabilistic graphics models and vibrational Bayesian techniques (Jamil et al. 2022)(Chen et al. 2019; T. Chen et al. 2021). Mean and variance for latent space distributions are incorporated during VAE compression. VAE produces better results than straightforward AE. VAE was utilized by several researchers to compress still images. The authors

of (T. Chen et al. 2021)(Chen et al. 2019) showed how to compress images using VAE with a nonlinear transform and uniform quantize. The model was trained using the Challenge One Learned Image Compression (CLIC) dataset. Due to the large amount of training data, the model was difficult. The mean-squared error (MSE) or perceptual metrics like MS-SSIM are typical loss functions to quantify the distortion between the original and decompressed images. Some writers use cutting-edge methods including generalized divisive normalization (GDN) layers, multi-scale decompositions, and progressive encoding/decoding methodologies.

The usage of generative adversarial networks (GANs) is a popular technique for unsupervised learning of generative models for challenging distributions [(Van Den et al. 2016). Despite stability issues, it was shown that they could create images that were harder and more realistic than those made by earlier techniques (Goodfellow et al. 2014; Im et al. 2016)and scale to resolutions of 10241024px (Wu et al. 2020) for particular data sets. Conditional GANs which have achieved outstanding results for image-to-image translation (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019; Knyaz, Kniaz, and Remondino 2019a; Wu et al. 2020) on a variety of data sets (such as maps to satellite images) and can translate images at resolutions of up to 1024 x 2048 pixels (Wu et al. 2020), are another area that has undergone significant development.

Deep learning had a significant impact (Jamil et al. 2022)(Torfason et al. 2018a)on every field of research. There was a revolution in image processing due to the compiling property of feature extractions in convolutional neural networks (CNNs). Many deep learning architectures (Jamil et al. 2022) were proposed to compress images. These architectures were categorized based on the type of deep learning model used.

2.3. Image-to-Image Translation

Image processing and computer vision have made significant progress when it comes to image-to-image translation. The goal of mage-to-image translation to transfer images from one domain to another while maintaining the content representations. For example, it can change a face photo from grayscale to color or from a photograph to a cartoonish representation. Examples of image-to-image translation transfer include pose estimation, image synthesis, segmentation, picture reconstruction, and style transfer (Tschannen et al. 2018). To create the desired output image, the process involves learning a mapping between the input and output images.

The goal of image-to-image translation is to change an input image from one form to another, such as from black and white to color or from a photo of a face to a cartoon-like drawing. It can be used for various tasks, such as estimating poses, creating images, segmenting images, restoring images, and transferring styles (Tschannen et al. 2018). The method is to learn a mapping between the input and output images and then use that mapping to produce the desired output image.

Image-to-image translation has been improved by GAN-based algorithms and produced impressive results (Emami et al. 2021a). Pix2pix (Knyaz, Kniaz, and Remondino 2019b used cGAN to translate a source image to an output image using matched images from the source and target domains. CycleGAN (Jay et al. 2017 solved image-to-image translation problems without paired examples.

In order to produce the desired output when the resultant image is created by the compression and subsequent decompression of the input image, the technique requires learning about the relationship between the input and output images. By precisely replicating the distribution of the target domain, a generative model can produce translated images that unmistakably resemble samples from the target domain (Zhang and Dong 2020).

Since image-to-image translation tasks necessitate understanding the links between multiple image-to-image domains, the effectiveness of generative models directly depends on how well these relationships are represented in order to obtain the appropriate results. The generative model considers that (Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019; Tschannen et al. 2018)image generated and discriminator network identify the feck image generated by generator from real image. In (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019; Iwai et al. 2020) image compression GAN model image compression and decompression inside of generator model.

2.4. Generative Adversarial Network

GAN (Goodfellow et al. 2014; Im et al. 2016) consisted of two neural networks, Generator G and Discriminator D that were trained together by playing a zero - sum game in which the generator tried to fool the discriminator with fake input. Therefore, generative means creating deceptive data, and adversarial means having a competition between the discriminator. Both models improved by adjusting their weights during training until they reached Nash equilibrium or zero-sum. The two

interacting networks, the generator and the discriminator, competed in an adversarial way, with the discriminator trying to distinguish fake samples from real input data. In this work (Minnen and Singh 2020) they describe the conditional version of generative adversarial networks, which can be created by simply providing the data, y that they want to condition on, to both the generator and the discriminator.

GAN training usually has two stages. First, freeze the (Goodfellow et al. 2014) (Im et al. 2016) generator and train the discriminator, postponing generator training and allowing only a forward pass with no backward propagation. Second, freeze the discriminator and teach the generator to produce more realistic results that would trick the discriminator. The generator and discriminator are often represented by multilayer networks with convolutional and/or fully connected layers. The generator and discriminator networks have to be differentiable, but they do not have to be directly invertible. It was applied to many applications like speech synthesis, image-to-image translation, image super-resolution, music generation, and video synthesis. The basic framework of GAN is shown in the following Figure 3.

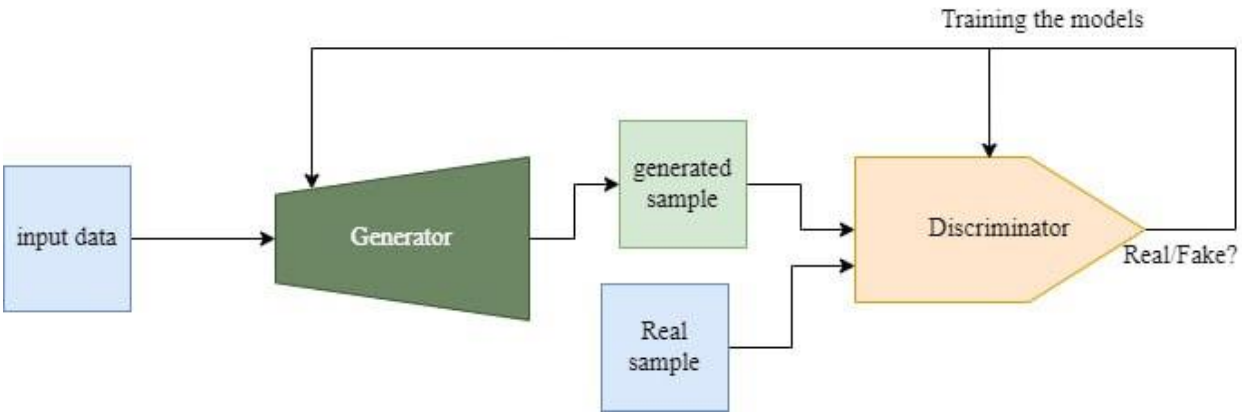

Figure 2. 3 GAN Architecture (Goodfellow et al. 2014)

Generative Adversarial Networks (GANs), a new method for image compression, have just been suggested. GANs are a subclass of deep neural networks that can produce fresh data samples that resemble an existing dataset. GANs were also utilized for image compression (Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019; Galteri et al. n.d.; Iwai et al. 2020; Mentzer et al. 2020; Quan et al. 2018; Tschannen et al. 2018) authors used the architecture of GAN algorithmic to handle the quality of image and compresses in low bit rate. The above (Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019; Iwai et al. 2020) propose and studies a

novel problem of distribution-preserving lossy compression, which aims to optimize the quality of image generation.

This work (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019) has two generative models, generative compression (GC) and selective generative compression (SC). This GC preserves the overall image content while generating structure of different scales, such as leaves or windows. The above authors are motivated by this problem by recent advances in extreme image compression, which can produce realistic and artifact-free reconstructions at very low bitrates by learning to synthesize image content. The paper shows that the system produces visually pleasing results at bitrates where previous methods fail and show strong artifacts. The paper also provides a user study that confirms that the system outperforms state-of-the-art methods in terms of visual quality and bitrate savings. The authors also provide theoretical and empirical analysis of their methods on two standard GAN datasets, CelebA and LSUN bedrooms, and demonstrate that their methods can effectively learn a compression system that respects the distribution of the original data at all rates.

2.5. Related Work

In recent years, several researchers have proposed methods for GAN-based image compression, this researcher was (Agustsson et al. n.d.-a) (Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019; Iwai et al. 2020; Mentzer et al. 2020; Tschannen et al. 2018; Wu et al. 2020) provided suggestions on how to compress full-resolution images while aiming for a low bit-rate of less than 0.1 bpp.

This work (Torfason et al. 2018b) offers a novel notion of performing image compression tasks compress the size of images in low bit rate. In this works used the generated by deep neural network (DNN)--based image compression techniques, thereby removing the need for RGB image decoding. According to the paper, this technique has the potential to reduce computational overhead, improve image quality, and use the knowledge collected by compression networks for inference. The study shows that doing image classification using compressed representations produces approximately comparable accuracy as performing image classification using decompressed pictures. Furthermore, it shows that semantic segmentation accuracy using compressed representations is comparable to that of decompressed images at moderate compression rates while outperforming them at aggressive compression rates.

A work (Torfason et al. 2018b) image compression may have limits. The paper's first limitation is that it only addresses one image compression architecture and does not compare it to other learned compression approaches. The paper's second limitation is that it does not assess interpretation performance on increasingly difficult images interpretation tasks. The paper's third limitation is that it does not address to calculate between compression rate and original images.

In this author (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019) presented a system that used generative adversarial networks (GANs) extremely low bitrate images compress. (Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019) presented an extreme GAN-based image compression architecture. For image compression, they used two different kinds of modules. A deep convolutional auto encoder as the generator network and a deep convolutional discriminator network in the generator contain network encoder quantize and decoder network. This method was able to achieve high compression ratios while maintaining good image quality, and also has two module operation generative compression (GC) and selective generative compression (SC). GC preserved the overall image content while generating details that could not be stored, such as leaves or windows. SC completely generated parts of the image from a semantic label map, while keeping user-defined regions with high detail. (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019) showed that their system produced visually pleasing results at bitrates where previous methods failed and had strong artifacts. The model outperformed JPEG (Wallace 1991) and (Lee, Cho, and Kim 2019; Skodras et al. 2001) JPEG-2000 in terms of results. At low bit rates like 0.15 bpp, the visual quality of the reconstructed image was significantly better than JPEG and JPEG-2000. The authors introduced GAN-based image compression architectures in (Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019; Galteri et al. n.d.; Iwai et al. 2020; Mentzer et al. 2020; Tschannen et al. 2018; Wu et al. 2020) had achieved high compression efficiency.

In this work (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019) limitation was the method requires a semantic label map of the original image from the (SC) model. (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019) use off-the-shelf semantic segmentation networks to obtain the label maps, but they do not report how the quality of the label maps affects the compression results.

This author (Akutsu et al. 2020) proposed for Ultra-low bitrate learned image compression by selective detail decoding using generative adversarial networks. In this work (Akutsu et al. 2020) propose a novel method for image compression that achieves both low bitrate and high perceptual quality. The method uses two decoders: a main decoder that is trained by a conditional

generative adversarial network (GAN) and a selective detail decoder that is trained by a weighted multi-scale structural similarity (MS-SSIM) loss. The selective detail decoder only applies to important parts of the image, such as text and faces that are automatically detected by the method. The paper used (Nair et al. 2019)also introduces a causal attention module to the entropy estimator, which reduces the bits per pixel (bpp) by predicting the probability distribution of the quantized values more accurately. The paper uses a super-resolution-style encoder and decoder with (Sun et al. 2020) spatial attention modules to generate high-definition images. The proposed method describes the network configuration and the loss functions of the two decoders, as well as the entropy estimator with the causal attention module. The experiments show the results of the proposed method on various datasets, such as Kodak PhotoCD and CLIC2020, and compare them with other methods in terms of bpp, MS-SSIM, and peak signal-to-noise ratio (PSNR). While achieving an ultra-low bitrate. The paper also claims that the proposed method can output visually superior images that are closer to the ground truth than a GAN-based image compression model.

These studies demonstrate that GANs have potential for use in image compression. When designing GAN-based algorithms for compressing images, it is important to consider factors such as computational efficiency, compression ratio, and image quality. The selective detail decoder has a problem in capturing crucial image information in these investigations; the method is inadequate for compressing images with detailed structures or textures, such as natural scenes or paintings.

In (Iwai et al. 2020) the first module was generative compression, where no semantic label maps were required. in this (Iwai et al. 2020) paper, we proposed a novel image compression method that worked at extremely low bitrates below 0.1 bits per pixel (bpp). The method used generative adversarial networks (GAN) to reconstruct realistic and sharp images while avoiding undesirable noise and artifacts. We adopted a two-stage training strategy, where we first trained the encoder and decoder without GAN, and then fine-tuned the decoder with GAN. This stabilized the training and simplified the optimization. We also used (Wang et al. 2019) a network interpolation technique, where we merged two decoders with or without GAN into one decoder. This allowed the user to control the trade-off between perceptual quality and fidelity of the reconstructions, without re-training the models. Moreover, we utilized a Gaussian mixture model (GMM) (Viroli and McLachlan 2019) as the entropy model, instead of a single Gaussian distribution. This made the entropy model more flexible and adaptive to different image contents. We showed that the

proposed method could produce visually pleasing results at extremely low bitrates, compared to existing engineered and GAN-based codecs. We also conducted a user study to verify that the proposed method achieved better visual quality than (Agustsson, Tschannen, Mentzer, Timofte, et al. 2019; Tschannen et al. 2018) GAN-based method.

Table 2. 1 summary of related work

No	Paper	Dataset used and Size	Gaps
1	(Torfason et al. 2018b) Towards Image Understanding from Deep Compression without Decoding	ImageNet dataset (60,000 for training,10,000 for testing)	➢ Does not compress learned image compression. ➢ Does not address the association between compression ratio and quality of images.
2	(Akutsu et al. 2020) Ultra-Low Bitrate Learned Image Compression by Selective Detail Decoding	CLIC2020 Dataset used for training and Kodak Photo CD dataset used for testing	➢ Does not address the challenges of generating images with complex scenes, diverse objects, and fine details at low bitrates. ➢ And also, does not evaluate the perceptual quality of the generated images using metrics such as SSIM and PSNR.
3	(Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al. 2019):Generative Adversarial Networks for Extreme Learned Image Compression	Cityscapes data set: 2975 images for training and 20 images from the validation set for testing.	➢ While focusing on natural and street scenes, its effectiveness for other image categories such as faces, cartoons, and medical images remains unclear and problem of quality images through low bitrate compression ratio.

4	(Iwai et al. 2020)Fidelity-controllable extreme image compression with generative adversarial networks	COCCO dataset used for training 11870 and 40,000 for testing and also Kodak PhotoCD used for testing	➢ In terms of visual quality and user preference, it has removed the important feature affected during extremely low bitrate compression ratio. ➢ Limitation on the quality generated images.

As shown in Table 2.1, several types of research are conducted by different researchers to predict accurate attentive areas in image compression. However, their study has the limitation listed in table 2.1 like focusing only on low bit rate image compression and generate quality images. During the generative adversarial network-based most researcher in image compression minimized size of image very low bitrate and to generate quality of images used sematic level. And also, the use of inefficient methods to carry relevant information into image, compressed the image the major limitation of the previous study. Therefore, this study solves those problems by extracting important features that are known to contain major information to then reconstruct the ng image. And also, give priorities to that extracted information. Priorities of the feature is given depending on the importance of the feature, features that are less important have very high compression rate and little compress that important feature.

2.5.1. Baseline Work

We selected "Fidelity-controllable extreme image compression with generative adversarial networks" (Iwai et al. 2020) as baselines of our works based on the model performance and generated outcomes when compared to others. According to the Iwai research approach was structured around a two-stage training strategy. Initially, we conducted training for the encoder and decoder in the absence of GAN. Subsequently, we refined the decoder using GAN. This two-step training process notably improved training stability and streamlined the optimization process. Moreover, we utilized a network interpolation technique, where we combined two decoders - one integrated with GAN and another without - into a unified decoder, adding an innovative dimension to this approach.

This work (Iwai et al. 2020) was chosen as a key reference due to its demonstrated effectiveness in image compression and generation, all while without semantic labels. Subsequently, Iwai proceeded to refine the decoder using GAN, a step that significantly bolstered the stability of the training process and simplified the optimization procedure. Furthermore, we adopted a network interpolation technique, allowing us to seamlessly integrate two distinct decoders—one trained with GAN and one without GAN—into a cohesive and effective single decoder.

The authors showed that (Iwai et al. 2020)performed better than other state-of-the-art methods for low bitrate image compression in terms of both compression ratio and visual quality images. Generally, the combination of residual blocks, interpolation network, and two step training was a

powerful tool for image compression. It surpassed existing GAN-based image compression methods across a wide range of datasets and metrics, especially with complex-textured and structured challenging images (Iwai et al. 2020) this work in the low bitrate without the semantic level and was evaluated by quantitative metrics values of (PSNR) and (SSIM) better than other low bitrate image compression models.

CHAPTER THREE

3. MATERIALS AND METHODS

This section defines the data-gathering methods, software, and hardware tools utilized to complete the research successfully. To accomplish an objective of this research, the following approaches and methodologies will be employed. The technique used to achieve image compression extracts key features images and assigns priorities to the features based on GAN to compress images at a low bit rate and generate quality images. This chapter describes the mathematical formulation of the evaluation metrics needed to complete this work. The approach, dataset, and experimental techniques utilized to more fully answer the study questions are covered in this chapter.

In Figure 3.1, the illustration presents various phases, encompassing data collection, preprocessing, and utilization of datasets for both model training and testing, and ultimately, the assessment of the model's performance.

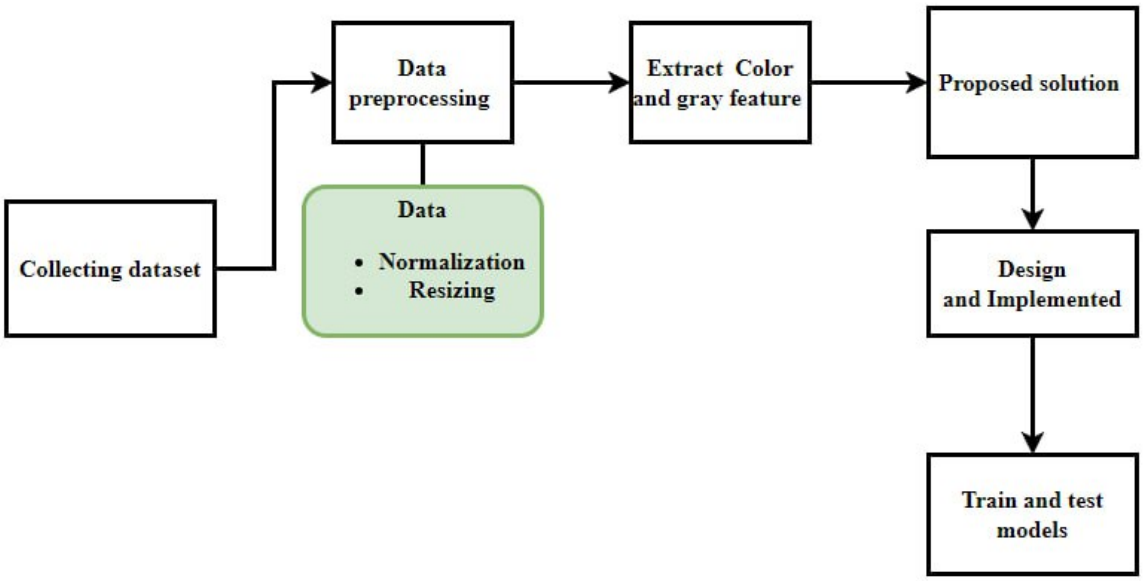

Figure 3. 1 Overall framework

3.1. Data Collection Methodology

The compression model we are proposed aims to compress images according to the key elements of properties of the image as a relevancy by given priorities. To accomplish this work, we have used a dataset prepared specifically for image compression models based on different properties.

We will be utilizing freely available online datasets such as COCCO (Lin et al. 2014). These datasets will be used for model training and testing.

The COCCO dataset [(Lin et al. 2014)**:** COCCO data sate was collected by the article Microsoft COCO: Common Objects in Context by Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick and Piotr Dollar was participated on collected COCCO dataset for object recognition and segmentation. The investigators team's reason to collect COCCO dataset was current datasets for object recognition are limited in several ways a small number of object categories have large number of categories but few instances per category.

This work (Lin et al. 2014) researcher proposes to address these limitations by creating a dataset that has a large number of object categories (91) that are common and recognizable by a 4-year-old. Offers per-instance segmentation masks for objects, which enable more correct localization and analysis of object shape and parts. The (Lin et al. 2014) collect images from Flickr using queries based on the selected categories and scene types (e.g., “kitchen”, “beach”, etc.). They filter out low-quality images, contain watermarks or logos, or are not relevant to the queries. Category detection: The writers use crowd workers to label each image with the categories that are present in it. They use a novel user interface that allows workers to quickly select multiple categories from a list or a hierarchy. The authors use crowd workers to annotate each instance of each category in each image with a segmentation mask. They use another novel user interface that allows workers to draw polygons around objects or refine existing masks.

Figure 3. 2 samples of COCCO dataset

The datasets were randomly divided into two groups: one for training the model and the other for testing. The training dataset was used to train the GAN-based image compression model and the testing dataset to evaluate its performance. Data pre-processing techniques such as resizing, normalization applied to the datasets before they were used for training and testing the model. The collected dataset was used to train and test the GAN-based image compression model, and the performance of the model was evaluated based on the compression rate and the visual quality.

3.1.1. Split Data for Training and Testing

The COCCO datasets were available on the Kaggle website which provides an open platform for researchers to share their object recognition. The website is specifically created to make the sharing of COCCO datasets from various vendors easier, by offering automatic conversion to the International instance and category of the dataset, parameter extraction, and thumbnail generation features. In the COCCO dataset categorize for training and testing, number of images for training 118,287 and for the testing 40,000 dataset and the size of original images 256x256. However, due to resource and time constraints that make it impractical to employ the entire dataset for training purposes, a careful selection process was undertaken. From the extensive pool of images resize image size 128x128 and selects 25,096 were chosen for training, and 7,500 images were earmarked for testing, providing a balanced and effective approach to harnessing the richness of the COCCO dataset while adhering to practical limitations, ultimately contributing to meaningful insights and robust model development.

3.2. Data Pre-processing

Image resizing: The dataset may contain images of varying sizes. Therefore, this should be transformed to the same size image. Image scaling is a critical preprocessing step in deep learning that enables input data to be exclusive to the network's size. The technique confirms the consistency of input image to the targeted size 128x128. This should be transformed to the same size as the image. Image scaling is a critical preprocessing step in deep learning that enables input data to be exclusive to the network's size. Resizing images to a fixed size also makes it easier to batch-process the images and input them into the model. However, if not done correctly, resizing images can main to loss of data and distortions

Normalization is a technique used to standardize the pixel values of images**.** This involved scaling the pixel values to a fixed range, such as [0, 1] to ensure that all images had the same dynamic range. Normalization helped prevent the GAN from learning biased features based on the pixel values of the input images. This technique also helped reduce the impact of outliers or extreme values in the pixel data.

3.3. Evaluation Metrics

In the proposed method to evaluate the performance of the proposed method of image compression led to the quality of compressing image data and the quality of generates image. Our work will use numerical evaluation metrics, measures Structural Similarity Index Measure (SSIM), and Peak signal-to-Noise Ratio (PSNR). These measurements can offer additional information that can contribute to the quality of generated image from the model.

The Peak Signal-to-Noise Ratio:

PSNR is a frequently employed metric for assessing the quality of image reconstruction (Sara, Akter, and Uddin 2019a). The power of the noise that distorts a signal's maximum possible value, or PSNR, is what determines how well that signal will be represented. The PSNR will be used to gauge an image's quality following reconstruction; a greater PSNR denotes a successful reconstruction. I and Y are the reference and filtered pictures of size, respectively, and MSE is calculated by averaging the squared intensities of the input and output image pixels.

PSNR (Sara, Akter, and Uddin 2019a) a commonly used metric for evaluating the quality of image reconstruction of images. PSNR is the ratio between the maximum possible value of a signal and the power of distorting noise that affects the quality of its representation. The PSNR will be used to measure the quality of an image after the reconstruction in which higher a PSNR indicates a good reconstruction. MSE is calculated by averaging the squared intensity of the input image and the output image pixels, where I and Y are the deference and filtered images of size respectively.

$$PSNR = -20 \log 10 \frac{(MAXF(X))}{\sqrt{MSE}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.\,.\ldots\ldots (3.1)$$

Structural Similarity Index Measure (SSIM)

The SSIM (Abdel-Salam Nasr, AlRahmawy, and Tolba 2017) metric measures how similar two supplied images are to one another. It carefully examines a pixel set, considering both low-level image properties, to more precisely identify the differences or similarities between the two images. The SSIM index, which is dependent on the assessment of three key elements—brightness, contrast, and the structural or correlation term—is crucial for the computation of a quality measuring metric. This index is created by multiplying these three components together (Sara, Akter, and Uddin 2019b).

Again luminance, contrast, and structure of an image can be expressed separately as:

$$L(x,y) = \frac{2\mu_x\mu_y + c1}{\mu_x^2 + \mu_y^2 + c1} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \quad (3.2)$$

Where c1 = (k1 + l) [2] is the non-negative regularization constant used to prevent instability when the mean is close to zero, l is the dynamic range's value, k1 is the small constant, and k1 is the small constant.

$$C(x,y) = \frac{2\delta xy + c2}{\delta_x^2 + \delta_y^2 + c2} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \quad (3.3)$$

Where l is the value of the dynamic range, k2 is the tiny constant, and c2 = (k2 + l)2 is the non-negative regularization constant used to prevent instability when the mean is close to zero.

$$S(x,y) = \frac{2\delta xy + c2}{\delta_x\delta_y} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \quad (3.4)$$

Where, µx average mean of x, µy average mean of y, δ_x^2 variance of x, $\delta_{(y)}^2$ the variance of y and δ_xy covariance of x and y, and c3 = c2/2

Luminance, represented by the letter "L" in this instance, is a measurement used to compare the brightness levels of two images. In order to emphasize the contrast between the lightest and darkest sections in these images, contrast, symbolized by the letter "C," is crucial. The local brightness patterns of the two images are quantified using the letter "S" to show both similarities and contrasts between them. Furthermore, all three of these constants were correctly defined by (Sara et al. 2019a) the structural similarity index method can be expressed through these three terms:

$$SSIM(x,y) = [L(x,y)]^\alpha \cdot [C(x,y)]^\beta \cdot [S(x,y)]^\gamma \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \quad (3.5)$$

In this context, α, β, and γ represent parameters employed to fine-tune the relative significance of the three components. The SSIM index ranges from -1 to 1, with 1 signifying complete similarity between the reconstructed image and the original image and -1 signifying complete dissimilarity. In actual applications, SSIM values typically range from 0 to 1, with larger values indicating better reconstruction quality.

3.4. Feature Extraction Techniques

Feature extraction processes are done at two different stages of our work. The first stage is during the dataset preparation and the second stage is during model training. Since extraction of both gray and color of images from dataset for our works. Grayscale images only have one channel that represents the luminance, or brightness, of the image, whereas RGB (Red, Green, Blue) images

(Gowda and Yuan 2019)have three color channels. During compression, the number of channels had a direct impact on the amount of data. Inherently, grayscale images needed less storage space and processing power than color images. Due to this distinction, grayscale photographs were a more practical option for applications requiring limited storage or bandwidth. Grayscale of images priority given focused on making them suitable for use where color information was less important. While color images offer great visual fidelity, they did an excellent job of representing a wide range of colors and tones. To maintain color accuracy and brightness during compression and decompression, color images were preferred when maintaining the visual appeal of the image was critical, such as in photography or graphic design. Grayscale images are often compressed more efficiently than RGB images due to their smaller data size. The simplicity of grayscale images can be exploited by compression techniques that reduce file size and preserve image quality.

3.4.1. Color Feature Extraction

We have created a new color feature extraction method for the extraction of color features. Ratio-based color feature extraction is the method used by the color feature extractor we created. The RGB channel sum is calculated first for each channel within a pixel of the image. The ratio of each channel to the total is then determined. Let f_c, f_g represents color and gray feature in mathematical expression of the feature extraction process from X images, X represents original images. Let S represent the summation elements calculated from summation of color channels perspective image X pixels.

$$S_{i,j} = \sum_{c=0}^{d} x_i \, j_c \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.6)$$

Where i= 0………. n, j = 0……m, c = 0………. d

n = represent rows of the image X

m = represent columns of the image X,

d = represent channels of the image X

Color feature extraction: let $S_{i,j}$ can be summation of the pixels of image X pixels

$$f\mathrm{c}(x_{i,j}) = \left[\frac{r}{s_{i,j}}, \frac{g}{s_{i,j}}, \frac{b}{s_{i,j}}\right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.7)$$

r, g, b represents RGB image colors

Gray feature extraction from color: which is mean of color channel of pixel p in image x

$$f\mathrm{g}(P_{i,j}) = \frac{\sum_{c=1}^{3}(P_C)}{3} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.8)$$

CHAPTER FOUR

4. PROPOSED SYSTEM DESIGN AND ARCHITECTURE

4.1. Introduction

In this chapter, we describe in detail proposed solution architecture of GAN-based image compression that prioritizes important feature as relevant. Using GAN, we explain the process of important feature prioritize image compression by diagrammatical and mathematical way. Architecture represents a compressive of overview of the steps involved our works including both diagrammatical mathematical formulas. The proposed solution of the image compression training different priority of features based different application area. The compression of images significantly reduces image size into a low bitrate, and result images have better-quality visualization generated image. And also reconstructs image details information based on given priorities. This section can be divided into three parts. The first part explained the detailed architecture the proposed model. The second parts describe learning model and learning function in detail. The last part explains how the model is trained and what pseudocode is utilized in training the model.

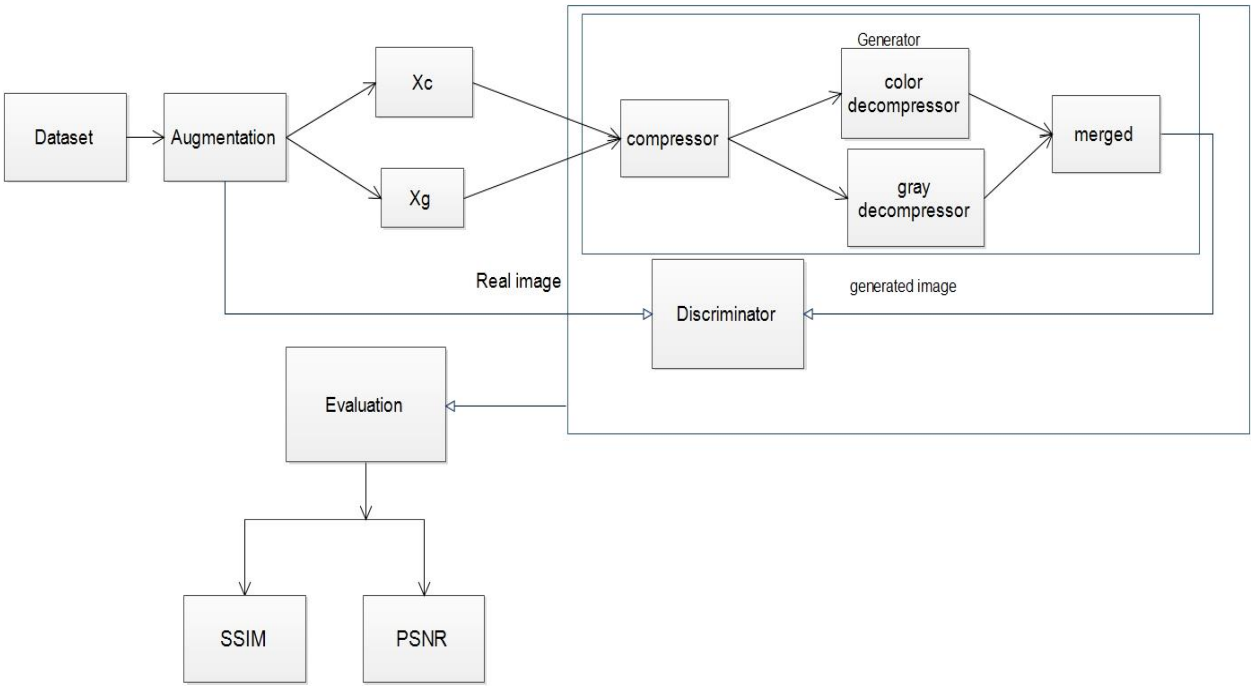

Figure 4. 1 The proposed model process over flow

Step 1: Data Preparation

We initiate by reading image dataset from specified directories and conducting data preprocessing, which involves resizing and normalizing the images. Additionally, we extract both color and gray features from these images, assigning priority to these features for image compression.

Step 2: Creation of Generator and Discriminator Models

We utilize TensorFlow and Keras libraries ability to the generator and discriminator models. The generator is designed to process the two prioritized image features, employing two decompressors to generate individual features. A merged feature use custom layer multiplication is employed to generate the final images. The generator's results, along with real images, serve as inputs for the discriminators, which are responsible for distinguishing between real and fake images.

Step 3: Utilization of Loss Functions

Our approach incorporates auxiliary loss and WGAN (Wasserstein GAN) loss functions to ensure that the generated images faithfully reconstruct the original image.

Step 4: Training

The next phase involves training using features extracted from the COCCO dataset images the generator networks compressor, decompressor and discriminator models.

Step 5: Testing

Once the models are trained, we employ the trained generator model, along with compressor and decompressor components, to compress images at low bitrates and generate images that closely resemble the input images.

Step 6: Evaluation

In the final step, we assess the performance of our model through quantitative measures such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), comparing the reconstructed images with the original ones to the quality and fidelity of the reconstruction.

4.2. Over View of Model Architecture

The proposed model learned image compression in very low bitrate based on important feature prioritize using GAN algorithm. In the proposed framework, we used a generative adversarial network (GAN) (Emami et al. 2021b; Goodfellow et al. 2014; Im et al. 2016) suggested deep learning model has two models called generator and discriminator. In the generator module we are proposed compressor and decompressor. The compressor module was compressing the feature according to given priorities, and also decompressor decompress feature in the same to

compressor. In generator model wen decompressor model decompress the feature the output put of decompressor is multiplayer by layer multiplication to generated the feature of images. The generators produce fake image after multiply all selected feature decompressed by layer multiplication the compressed information. This generated image targets to mimic a real image deceiving the discriminator. Generator continuously learn to update generated fake image similar to real image, while the discriminator experiences constant optimization to correctly evaluate the images generated by the generator. In discriminator network used Wasserstein and binary cross entropy loss to improve the quality of the generated images and reducing the vanishing gradient generative adversarial network.

Figure 4. 2 General Architecture

4.3. Generator Architecture

A generator models as seen in figure 4.2 are made up of two network compressor and decompressor modules, each module containing long-skip connected residual block. In compressor block applied down-sampling based on given priorities of the compressed feature, and in decompressor applied up-sample block. The decompressor model generates new samples from low bit-rate training data distribution that resemble real samples.

4.3.1. Compressor

The compressor model for image compression, which is an important component of learning-driven image compression based on properties prioritization. The compressor network in generator networks is adaptable to compress features based on their priority. The eternal block of the compressor model using four down sample blocks with skip block, each down-sample reduce dimensions and have a normalization technique. By providing the output dimensions and selecting between instance and batch normalization. Convolutional layers, normalizing layers, and an activation function were employed in the down-sample block function. This block was responsible lowering the input image's spatial dimensions while increasing the taken a number of filters to more compress the feature.

Pseudocode of compressor networks:

Down sample code has two 2D convolutional layers, Conv_layer1 is a Conv2D layer with the filter kernel_size_3, padding same, stride 2x2 and activation Relu parameters. Conv_layer2 is a Conv2D layer with the filter kernel_size_3, padding same, Skip connection is equivalent to Skip block (Filter, Input Tensor). merged = Element-wise addition of conv layer and skip connection

Returns merged

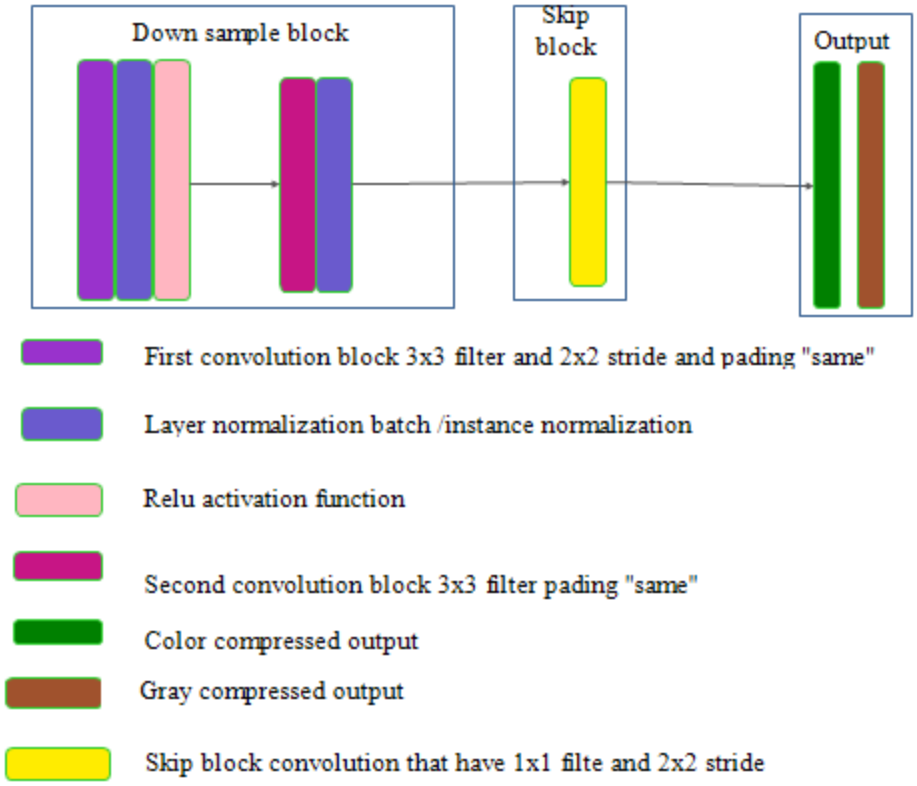

Figure 4. 3 down sample block internal part of the encoder

It offers flexibility by allowing the choice between (H. Chen, Wang, and Fan 2021)batch normalization and instance normalization for layer normalization. It was responsible for extracting essential features from the input image and lower dimensional representation. The last block function creates the final block in the network, responsible for reducing the spatial dimensions to a manageable size and extracting the desired features for the given task.

4.3.2. Decompressor model

The model of a decompressor models was designed for reconstructing the compressed image pass from low bit-rate. The decompressor model starts with a dense layer that takes the low-dimensional encoded representation as input and constructed the accepted the feature compressed according to priorities compress the feature. The decompressor architecture consisted of several layers reshape layer, which prepares it for up sampling, such as convolutional transpose normal convolutional layers were added to up sample decompressed feature. Each transpose layer doubled the spatial dimensions of the input. Convolutional layers were added after each transpose layer to perform feature decompress and enhance image reconstruction. Each convolutional layer performed a 3x3

convolution with Relu activation and a kernel initializer of ‘he normal’. The output of each convolutional layers was added element-wise to the corresponding up sampled layer using the Add layer. The final output was obtained using a 1x1 convolutional layer with sigmoid activation, which produced the decompressed feature of image according to give priority and output block module multiply decompressed feature by custom layer.

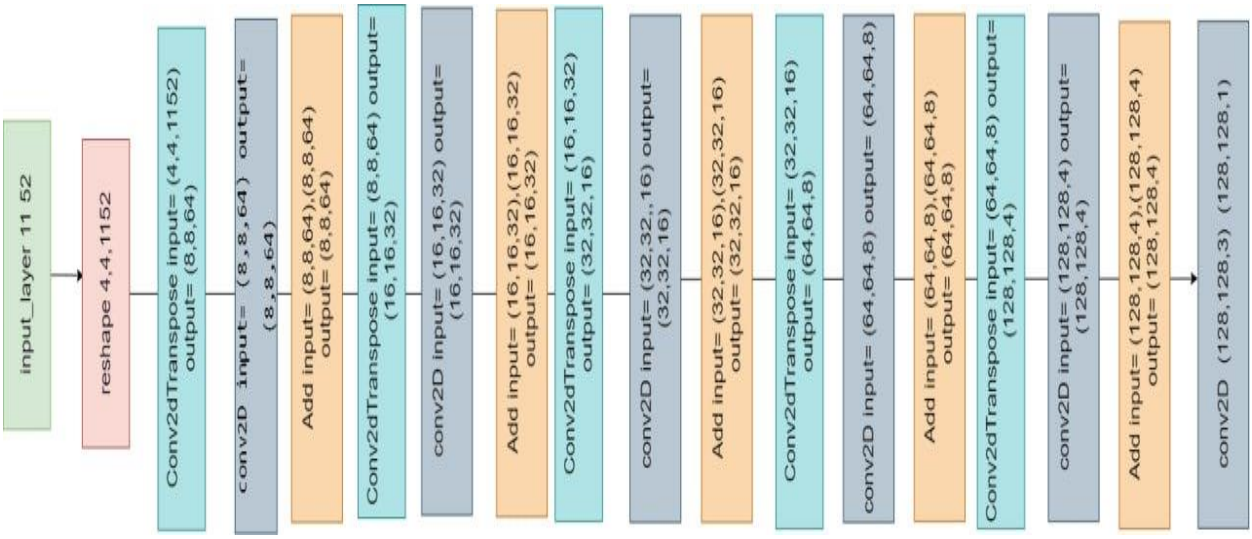

Figure 4. 4 decompression architecture

4.3.3. Output block module

Output block module was looks like a custom layer multiplication, output block which modifies the gray output and color output of the decompressor model by element-wise multiplication with an output of decompressor in each epoch value. Element-wise multiplication gray output is then combined with the color output to produce the final output of the model.

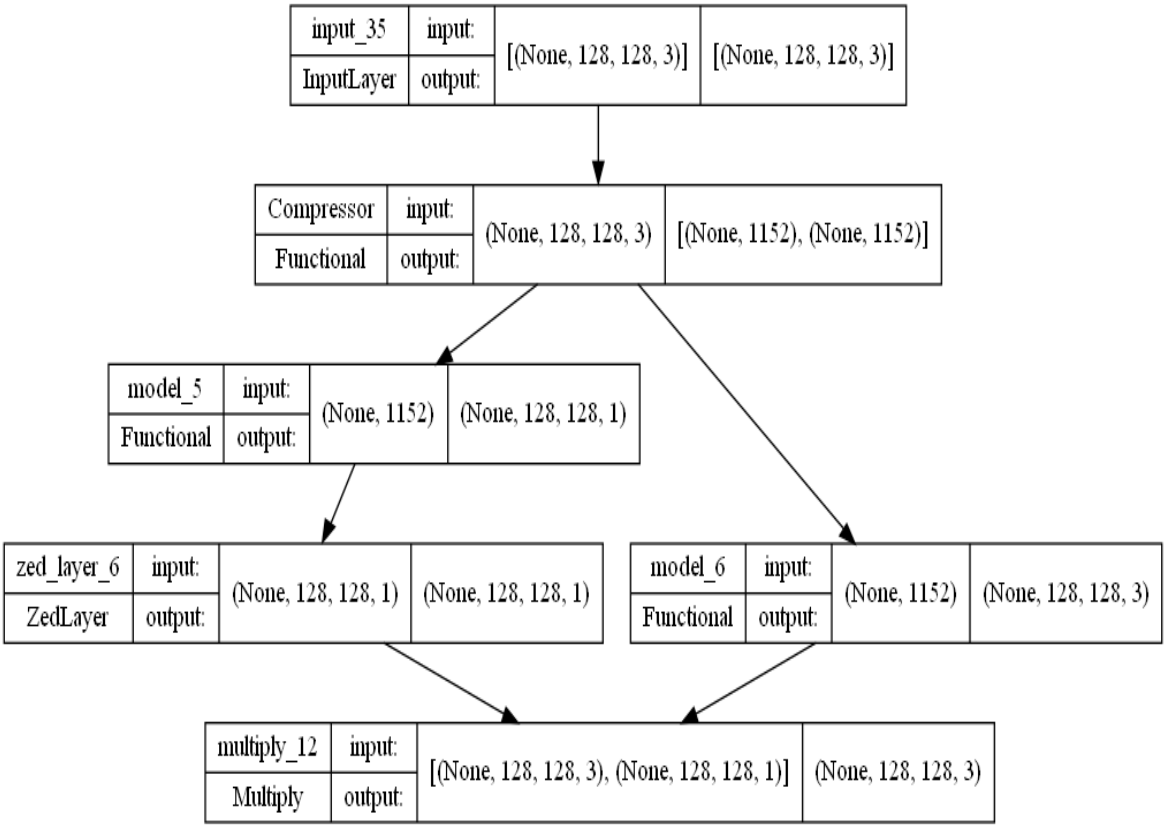

Figure 4. 5 output block of module

4.4. Loss Calculation Techniques

We describe the loss functions used for color and gray images in an image compression system. The loss functions are designed to quantify the deference between the reconstructed images and their corresponding ground truth images. By optimizing these loss functions, the compression system aims to generate compressed images that closely resemble the originals. The loss functions we used in this work consist auxiliary lossy in compressor and decompressor and Wassertainal lossy discriminator.

4.4.1. Auxiliary loss function

The auxiliary loss function is a common approach used in various generative models,(Du et al. 2023) such as Generative Adversarial Networks (GANs) or Variation auto encoders (VAEs). The resulting loss value was used to optimize the generator model during training. By minimizing this loss value, the generator model learned to generate images that were similar in content to the input images. In generally, the auxiliary loss function played a (Vafaeikia, Namdar, and Khalvati 2020) crucial role in training of GAN by hopeful to preserve the content of original images during

generation. This loss function aims to encourage the generator model to generate images that are visually similar to the original images. The loss function has two inputs: a real image and a generated image. Auxiliary lossy calculated the mean absolute difference used, and also enumerate the difference between the two images and provides a measure of how well the generator model generate the details and structure of the original image.

Auxiliary loss of network

$$gray\ loss = lg = \left||x_g, x'_g|\right| \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots .. (4.1)$$

$$color\ loss = \ lc = \|x_c, x'_c\| \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . . \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots ... (4.2)$$

Compressor loss

$$Compressor\ loss = \ lcm = \mathrm{lg} + \ lc + \frac{1}{3}\left(1 - l_{fake}\right) \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots .... (4.3)$$

Decompressor

$$gray\ decompressor\ loss = \ l_{dg} = \mathrm{lg} + \frac{1}{3}\left(1 - l_{fake}\right) \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \ldots . (4.4)$$

$$color\ decompressor\ loss = \ l_{dc} = \mathrm{lc} + \frac{1}{3}\left(1 - l_{fake}\right) \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots .. (4.5)$$

Total generator loss DC_g

$$Lg = \lambda 1 lg + \lambda 2 lc + \lambda 3\left(1 - l_{fake}\right) \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots .. (4.6)(\text{Du et al.} 2023)$$

Where: weight initialization of lambda

$$\lambda 1\ gray\ loss\ weight\ C\mathrm{g}, \mathrm{DCg} = 1$$

$$\Lambda 2\ color\ loss\ weight\ Cc,\ DCc = \frac{1}{3}$$

$$\lambda 3\ Cg, Cc\ DCg, DCc\ and\ Des\ wieght\ \ loss\ weitht\ = 100$$

$$Cg, DCg\ reperesenties\ gray\ feature\ compresor\ and\ decompressor$$

$$Cc, DCc\ reperesenties\ color\ feature\ compresor\ and\ decompressor$$

$$Des\ reperesentiesdescriminator$$

4.4.2. Discriminator loss function

In discriminator loss we used two deferent loss Wasserstein and binary cross-entropy loss is a measure of how well the discriminator model is able to distinguish between real and generated images. Binary cross-entropy (Guo et al. 2022; Vallecorsa, Carminati, and Khattak 2019)proposed for calculated by taking the negative log-likelihood of the correct classification for each image The WGAN loss(Arjovsky, Chintala, and Bottou 2017) is used to train the discriminator network to distinguish between real and fake images distribution predicted and real images. The loss function encourages the discriminator model to correctly classify real images as real and generated images as fake. The discriminator model is an important component of generative adversarial networks (GANs), which are used to generate new data that is similar to a given dataset. The generator model in a GAN generates new data that is then evaluated by the discriminator model. The generator model is trained to generate data that can fool the discriminator model into thinking that it is real data.

Wasserstein loss calculated as:

$$L_W GAN = E_{x\sim Pgen}[f(x)] - E_{(x\sim P_{real})}[f(x)] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.6)$$

Where P_{real} and P_{gen} are the true and generated probability distributions, respectively, and f is the critic function

Binary -cross entropy loss calculated as:

$$Loss\text{BCE} = \frac{1}{n}\sum_{i=1}^{n}(y.\log(y') + (1-y).\log(1-y')) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.7)$$

We train our models prioritize of relevant properties improve low bitrate image compression using the combination of the WGAN loss and the auxiliary loss discriminator and generator network. The auxiliary loss assesses how effectively the generator network can maintain the semantic information of the original images, whereas the WGAN loss measures how well the discriminator network can distinguish between genuine and created images. To increase the quality of the generated images, we update the generator network during training to reduce the auxiliary loss. In order to maximize the WGAN loss and increase the compression ratio, we additionally update the discriminator network. To balance image compression and quality, we often used both Wasserstein loss and auxiliary loss.

4.5. Discriminator Network

In our model's architecture, the (Im et al. 2016) discriminator plays a crucial role in evaluating the authenticity of generated images. It uses a standard discriminator design with an integrated generator network that focuses on extracting relevant features and prioritizing image compression. This integration enables the discriminator to differentiate between real and generated images, which helps evaluate image quality and improve the generator's performance during the training phase. By employing a binary classification task to distinguish real from fake images, the discriminator provides essential feedback to the generator, guiding it towards generating higher-quality images. The discriminator model includes various layers, such as convolutional layers, zero-padding layers, batch normalization layers, and dense layers, which enable it to learn and analyze the input images' features effectively. The loss functions used in training the discriminator, including Wasserstein loss and binary cross-entropy, contribute to achieving the primary objective of successful image differentiation.

The standard discriminator has several advantages within our model architecture. Firstly, it enhances the generator's training by providing real-time feedback based on the discriminator's assessment of image authenticity. This feedback mechanism guides the generator to generate images that closely resemble real images, improving overall image quality. Additionally, integrating a standard discriminator promotes feature extraction and prioritizes relevant features crucial for successful image compression. By focusing on these significant features, the discriminator optimizes the learning process, leading to higher-quality compressed images. Moreover, the discriminator's ability to distinguish between real and generated images aids in preventing mode collapse.

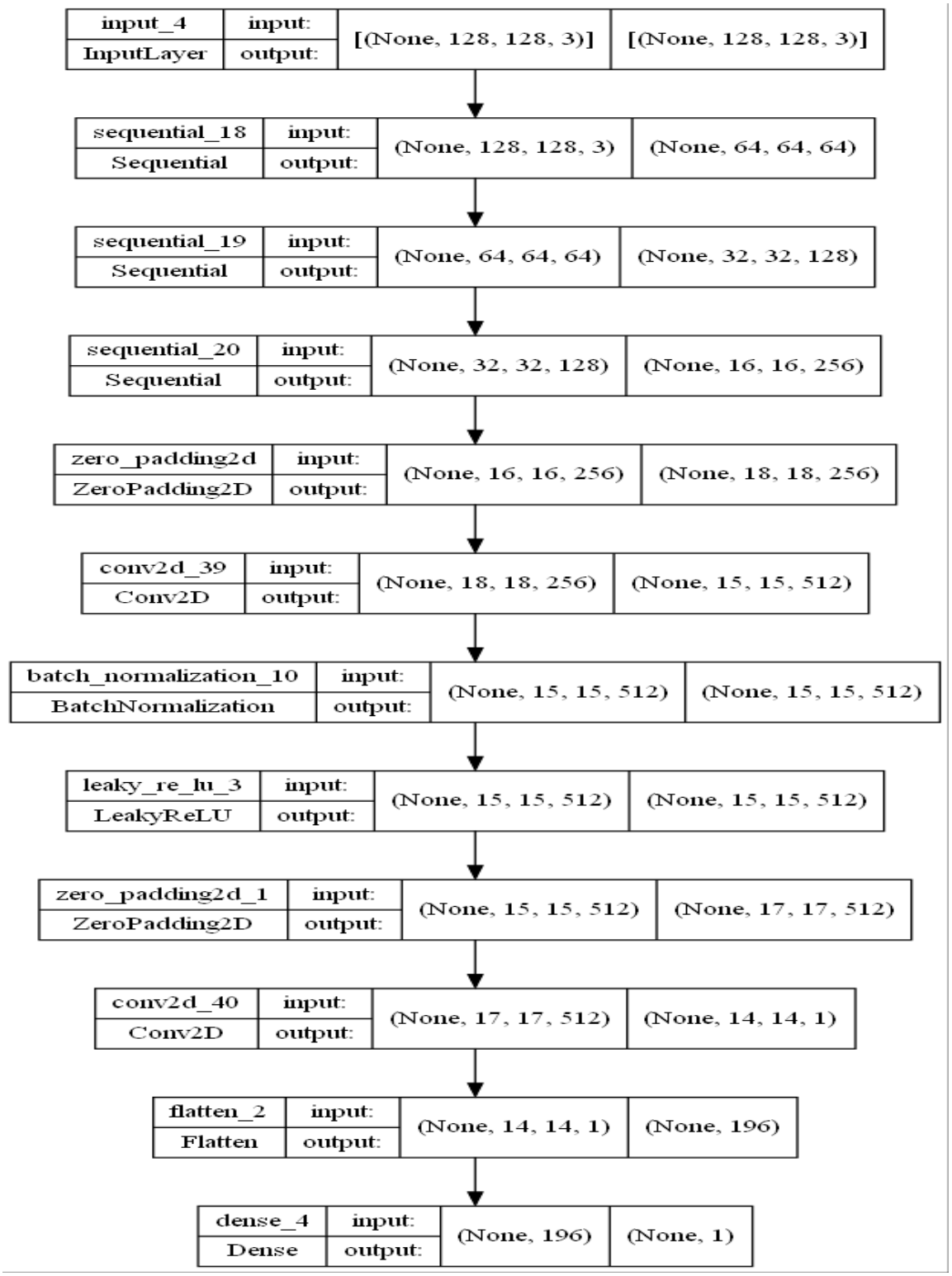

Figure 4. 6 discriminator network architecture

The input of discriminator model is two fake or generated image and "real image" represented the ground truth image. These inputs were combined to create the single tensor "x," which was formed, this merged tensor was processed via a number of layers, each of which increased the number of channels while decreasing the spatial dimensions of the feature maps. This made it possible for the discriminator to collect increasingly high-level and abstract properties.

4.6. Training Pseudocode

Algorithm1: Propose Relevant properties prioritized-G and Standard Discriminator -D GAN algorithm

1. **Initialize** *generators network G contains compressor and decompressor Xg, Xc feature of images extracted and discriminator D*
   Priority given to the feature Xg, Xc represents gray feature and color feature respectively.
   Two decompressors used to decompress feature according to given priority produce X'g, X'c merged layer multiplication added in generator part take decompressed feature.
   Standard discriminator at discriminator sides
2. **Input:** *Xg, Xc from COCCO images A (1……n) m number of iterations*
3. ***For** numbers of epochs **do***
4. ***For** n train-steps **do***
5. ***Update** the discriminator D using*

$$\nabla_{\theta d} \frac{1}{A} \sum_{a=1}^{n} [\, E_{\{x \sim P_{gen}\}} [f(X)] - E_{\{x \sim P_{real}\}} [f(X)] \,]$$

6. ***Update** In the generator compressor and decompressor models*

$$\nabla_{\theta g} \frac{1}{A} \sum_{a=1}^{n} \sum_{i=0}^{m} [||G(f(X'g, X'c) - (Xg, Xc)||] + \sum_{y \sim pdat(y)} [||Dc(f(y) - y||] \;\;]$$

7. ***End for***
8. ***Testing** using the cocco dataset testing data for model evaluation*
9. ***End for***
10. **Output** *generated quality image from low bitrate*

Bold area was update

CHAPTER FIVE

5. IMPLEMENTATION OF PROPOSED SYSTEM

5.1. Overview of Implementation

This chapter explains the proposed model's implementation, which includes tasks such as environment setup, data preprocessing, and the training RPPGAN process. Furthermore, as part of the implementation process, the experimental class is covered in this chapter. It also offers explanations of code screenshotted and attached in appendices and other information about implementation is discussed and included in the body of this chapter.

5.2. Working Environments

5.1.1. Hard work tools

This section discusses the environmental setup and hardware tools required to implement them. The proposed solution's working environments for as relevant properties, important feature of image extraction and give prioritize image compression in low bitrate using GAN. This includes the study's hardware and software. The hardware used a high-performance computer included CPU and GPU for efficient processing of data. The provided environments below facilitate the effective development, training, and evaluation of the proposed method for image compression with critical feature prioritization at low bitrates using GANs.

- ✓ Laptop:
  - Operating System: Windows 11 64-bit OS, x64-based processor
  - Microsoft Office 2021 version
  - Processor: Intel® Core™ i3-5005u CPU @ 2.20GHz 2.2GHz
  - Installed memory (RAM): 8.00 GB
- ✓ Desktop:
  - Computer System and Processor: Windows 10 Home, Intel® Core™ i7-8700 CPU @ 3.20 GHz (12 CPUs) - 3.2 GHz
  - Installed Memory (RAM): 16.00 GB
  - System Type: 64-bit OS, x64-based processor
  - Graphics Memory: Intel® HD Graphics 520/NVIDIA Corporation GeForce RTX 3060 with 12 GB RAM

The environments listed below allow for the efficient creation, training, and assessment of the suggested method for critical feature prioritized image reduction low bitrate GAN.

5.1.2. Software tools

To conduct the study through coding, a variety of writing software and coding tools will be employed. The following list provides descriptions of these tools for better understanding:

EdrawMax: a software tool that is used for the creation, presentation, and interpretation of design concepts. EdrawMax will be used for designing purposes in the proposed system

Anaconda: Anaconda is an open-source platform that makes maintaining software packages and environments straightforward, making it an indispensable tool for academics who use a range of programming languages and libraries. It includes a package manager called conda, which helps researchers to swiftly install, update, and manage various software packages, libraries, and dependencies, guaranteeing reproducibility and consistency in research projects. Building isolated, customizable environments is one of Anaconda's key characteristics, allowing academics to work separately on multiple projects with different software requirements. It provides access to a wide range of pre-built packages and data science tools, such as Jupyter Notebook and NumPy, and serves as a one-stop shop for research projects including data analysis, machine learning, and scientific computing.

Jupyter Notebook is a widely used tool in research of deep learning and machine learning software development. Jupyter also provides a web-based interactive environment that lets scholars combine text, code, and visualization in single document. One of its key features is that it is compatible with various programming languages, such as Python and R, which makes it ideal for data analysis and machine learning research. With Jupyter Notebooks, researchers may collaborate, discuss, and easily record their work. Since it enables repeated study and testing by allowing code cells to function independently, it is an essential tool for academics working in fields like data science and computer science.

TensorFlow: TensorFlow is an open-source machine learning framework developed by Google that is widely used in research due to its flexibility and scalability in building and training deep learning models tensorflow provides both high-level and low-level APIs so that researchers can choose the level of abstraction that best satisfies their goals. It can handle both large datasets and complex neural network topologies thanks to GPU acceleration and efficient computation graph execution. Because to tensorflow strong ecosystem, active community, and visualization tools like

tensorboard, researchers in machine learning, natural language processing, computer vision, and other fields rely done on it.

5.3. Environmental Setup

This study to accomplish use different software and Integrated Development Environments (IDEs) that support popular machine learning frameworks such as TensorFlow.

Table 5. 1 Environment Setup

No	Tools Name	Type of Tools	Properties		Description
			Name	Value	
1	TensorFlow		Version	2.8	Deep learning framework
2	Anaconda	IDE	Version	3.0	Packing python and Jupyter notebook
3	Jupyter Note Book	Editor	Version	6.5.3	For code editing purpose
4	Python	Programming language	Version	3.8	Compiling purpose and code management

5.4. Layers and Blocks Implementation

5.4.1. Down sampling block implementation

For down sampling we have developed function or method the returns object according to the parameter passed to it. This method or function is used to handle code complexity and redundancy. This bock is used by function used to define generator and discriminator.

```
def downsample_block(filter_,norm='batch'):
    model=tf.keras.models.Sequential()
    model.add(tf.keras.layers.Conv2D(filter_, (3,3), padding = 'same', kernel_initializer='he_normal', strides = (2,2)))
    if norm=='batch':
        model.add(tf.keras.layers.BatchNormalization())
    else:
        model.add(InstanceNormalization(axis=3))
    model.add(tf.keras.layers.Activation('relu'))
    model.add(tf.keras.layers.Conv2D(filter_, (3,3), kernel_initializer='he_normal', padding = 'same'))
    if norm=='batch':
        model.add(tf.keras.layers.BatchNormalization())
    else:
        model.add(InstanceNormalization(axis=3))
    return model
```

5.4.2. Skip block implementation

We have developed function skip block method that takes parameters used to adding properties, between two down sample block to be created and returned, for the next down sampling block.

```
def skip_block(filter_):
    model=tf.keras.models.Sequential()
    model.add(tf.keras.layers.Conv2D(filter_, (1,1), strides = (2,2), kernel_initializer='he_normal'))
    return model
```

5.4.3. Get compressor block Implementation

In this get block of compressor model, important feature of image extraction and give priorities to the feature to compress. The networks of the models were difference to each other depending on given priorities of the feature.

```
def get_block(x,out_dim,norm='batch'):
    normal_blocks = []
    skip_blocks = []
    filters = [32,64,128,256,512]
    last_b = last_block(out_dim)
    for filt in filters:
        normal_blocks.append(downsample_block(filt,norm))
        skip_blocks.append(skip_block(filt))
    for block,skip in zip(normal_blocks,skip_blocks):
        normal_x = block(x)
        x_skip = skip(x)
        x = tf.keras.layers.Add()([normal_x, x_skip])
        x = tf.keras.layers.Activation('relu')(x)
    last=last_b(x)
    return last
```

5.4.4. Compressor model implementation

In this model as relevant properties, important feature of image extraction and give priorities to the feature to compress. The networks of the models were difference to each other depending on given priorities of the feature.

```
def compressor(out_dims,name='Compressor'):
    inp_0=tf.keras.layers.Input(shape=[128,128,3])
    out_puts=[]
    norms=['inst','batch']
    for out_dim,norm in zip(out_dims,norms):
        out_puts.append(get_block(inp_0,out_dim,norm))
    model=Model(inputs=inp_0,outputs=out_puts,name=name)
    #model.compile(optimizer='adam',loss='mse')
    return model
```

Table 5. 2 Content of compressor model

Number layer	Content of compressor model
1	Conv2d(128,128,3), filter(3,3), padding=”same”, stride= (2,2), Normalization= BatchNormalization, activation=”relu”
2	Conv2d(64,64,32), filter(3,3), padding=”same”, Normalization= BatchNormalization, activation=”relu”
3	Conv2d(64,64,32), filter(3,3), padding=”same”, stride= (2,2), Normalization= BatchNormalization, activation=”relu
4	Conv2d(32,32,64), filter(3,3), padding=”same”, Normalization= BatchNormalization, activation=”relu”
5	Conv2d(32,32,64), filter(3,3), padding=”same”, stride= (2,2), Normalization= BatchNormalization, activation=”relu”
6	Conv2d(16,16,128), filter(3,3), padding=”same”, Normalization= BatchNormalization, activation=”relu”
7	Conv2d(16,16,128), filter(3,3), padding=”same”, stride= (2,2), Normalization= BatchNormalization, activation=”relu”

8	Conv2d(8,8,256), filter(3,3), padding=”same”, Normalization= BatchNormalization, activation=”relu”

5.4.5. Decompressor implementation

Implementation of decompressor as relevant properties, important feature of image extraction and give priorities to compress.

```
def decompressor(out_dim,ch=1,name='Decompressor'):
    in_x_0=tf.keras.layers.Input(shape=out_dim)
    shape_=4*4*out_dim
    x = tf.keras.layers.Dense(shape_,activation='relu')(in_x_0)
    x = tf.keras.layers.Reshape((4,4,out_dim))(x)
    u7 = tf.keras.layers.Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(x)
    c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(u7)
    c7 = tf.keras.layers.Add()([u7, c7])

    u8 = tf.keras.layers.Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c7)
    c8 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(u8)
    c8 = tf.keras.layers.Add()([u8, c8])

    u9 = tf.keras.layers.Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')(c8)
    c9 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(u9)
    c9 = tf.keras.layers.Add()([u9, c9])

    u10 = tf.keras.layers.Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')(c9)
    c10 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(u10)
    c10 = tf.keras.layers.Add()([u10, c10])

    u11 = tf.keras.layers.Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')(c10)
    c11 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(u11)
    c11 = tf.keras.layers.Add()([u11, c11])

    out_puts = tf.keras.layers.Conv2D(ch, (1, 1),  kernel_initializer='he_normal', activation='sigmoid')(c11)
    model=Model(inputs=in_x_0,outputs=out_puts)
    #model.compile(optimizer='adam',loss='mse')
    return model
```

Table 5. 3 Content of decompressor model

Number Layer	Content of decompressor model
1	Conv2DTranspose (4,4,1152), filter (2,2), stride= (2,2), padding='same'
2	Conv2D (8,8, 64, filter (2, 2), activation='relu', kernel initializer='he_normal', padding='same')
3	Conv2DTranspose (8,8,64), filter (2,2), stride= (2,2), padding='same'
4	Conv2D (16,16, 32), filter (2, 2), activation='relu', kernel initializer='he normal', padding='same')
5	Conv2DTranspose (16,16,32), filter (2,2), stride= (2,2), padding='same'
6	Conv2D (32,32, 16), filter (2, 2), activation='relu', kernel initializer='he normal', padding='same')
7	Conv2DTranspose (32,32,16), filter (2,2), stride= (2,2), padding='same'
8	Conv2D (64,64, 16), filter (2, 2), activation='relu', kernel initializer='he normal', padding='same')
9	Conv2DTranspose (64,64,16), filter (2,2), stride= (2,2), padding='same'
10	Conv2D (128,61284, 16), filter (2, 2), activation='relu', kernel initializer='he normal', padding='same')

5.4.6. Discriminator Implementation

We used a multilayer discriminator for this works as relevant properties, important feature of image extraction and give priorities to compress and decompress to generate high quality images in the discriminator part of GAN to improve the performance of the discriminator. Here is the implementation

Table 5. 4 Content of discriminator model

Number layer	Content of discriminator model
1	Conv2d(128,128,3), filter(3,3), padding=”same”, stride= (2,2), activation=”relu
2	Conv2d(64,64,64), filter(3,3), padding=”same”, stride= (2,2), activation=”relu
3	Conv2d(32,32,128), filter(3,3), padding=”same”, stride= (2,2), activation=”relu
4	Zero_padding2d1(16,16,256), filter(3,3), , activation=”relu”

5	Conv2d(18,18,512), stride= 1, filter(3,3), , activation=”relu”
6	Batch_normalization(15,15,512), filter(3,3), activation=”relu”
7	leakyRelu(115,15,512),
8	Zero_padding2d2(15,15,512), filter (3,3), activation=” relu”
9	Conv2d (17,17,512), stride= 1, filter (3,3), activation=”relu”
10	Flatten_14(14,14,1)

5.4.7. Pseudocode Implementation

The implementation of the training step of our models.

```
opt_enc=tf.keras.optimizers.Adam(learning_rate=0.0001)
opt_dec_g=tf.keras.optimizers.Adam(learning_rate=0.0001)
opt_dec_c=tf.keras.optimizers.Adam(learning_rate=0.0001)
@tf.function
def tran_steps(encoder,decoder_g,decoder_c,disc_model,data_x,saving=False,l1=10,l2=10):

    with tf.GradientTape() as tape_encoder,tf.GradientTape() as tape_decoder_g,tf.GradientTape() as tape_dencoder_c
        g_enc_y,c_enc_y=encoder(data_x,training=True)
        pred_img_g=decoder_g(g_enc_y,training=True)
        pred_img_c=decoder_c(c_enc_y,training=True)
        img_pred=(pred_img_g*3.0)*pred_img_c
        fake_label=disc_model(img_pred,training=False)
        loss_bce=bce(tf.ones_like(fake_label),fake_label)
        loss_mse=mse(data_x,img_pred)
        loss=(l1*loss_bce)+(l2*loss_mse)
        loss_enc=(1/3)*loss
    gradients_enc = tape_encoder.gradient(loss_enc, encoder.trainable_variables)
    opt_enc.apply_gradients(zip(gradients_enc, encoder.trainable_variables))
    gradients_decoder_g = tape_decoder_g.gradient(loss_enc, decoder_g.trainable_variables)
    opt_dec_g.apply_gradients(zip(gradients_decoder_g, decoder_g.trainable_variables))
    gradients_opt_dec_c = tape_dencoder_c.gradient(loss_enc, decoder_c.trainable_variables)
    opt_dec_c.apply_gradients(zip(gradients_opt_dec_c, decoder_c.trainable_variables))
    return (1/3)*loss_enc,img_pred
```

5.5. Hyper-parameter Configuration

The hyper-parameters variables are used to determine the network structure and how the networks are trained. Essential parameter was established, such as batch size, number of epochs, optimization, learning rate, and loss weight, is referred to as a hyper-parameter. These parameter variables were modifiable and straight touch how well a model train. Adam optimizer with two learning-rate $=10^4$ for discriminator learning and learning-rate $=10^3$ for learning generator

network used to update the network weight. Optimizers play a crucial role in adjusting the weights and biases of hyper-parameters within a model. When aiming to minimize losses effectively, a sufficiently high learning rate is essential. Adam, a widely recognized optimizer, is an amalgamation of two well-established stochastic gradient descent algorithms, AdaGrad and RMSProp, as proposed by (Kingma and Ba 2014) As highlighted in the research by (Hassan et al. 2023) Adam is valued for its user-friendly nature, computational efficiency, and memory efficiency in the training process. The learning rate, often referred to as the step size, governs the extent to which weights are modified during training, thereby influencing the optimization process. In the scope of this study, we adhere to parameters for weight loss and learning rate akin to those specified in (Iwai et al. 2020). The baseline model undergoes training with a batch size of 4, iterated over 500,000 cycles. The hyperparameter details for our model are outlined as baseline works, encapsulating crucial information regarding weight adjustment and learning rate optimization techniques. Our model undergoes training with a batch size of 2, iterated over 12548 cycles and Adam optimization with learning rate selection. The hyperparameter details for our model are outlined in the table below, encapsulating crucial information regarding weight adjustment and learning rate application.

Table 5. 5 Hyper-parameter values

Hyper-parameter	Values
Batch size of the training	2
Epoch numbers	100
Iteration numbers	12548
Optimizer	Adam optimizer
Learning Rate	Generator 0.0001, discriminator 0.00001

5.6. Experiment Class

The work describes the experiments conducted to, important feature of image extraction and give priorities as relevant properties to compress. The improvement of our work to minimize image size in low bit rate and reconstruct the quality image. There are four separate experiments carried out as shown in tables 5.6. The implementation of the baseline work constitutes the first experiment. In the second experiment, extract two features gray and color, this two-feature was important to contain more information of image, so give equal priority train first experiment. These two features

were incorporated in the last layer of generator decoder network output block. Output block custom layer multiplication with the addition of instance normalization or batch normalization in the generator encoder and decompressor network while keeping auxiliary loss, which is MSE loss.

In the third experiment, similar to second experiments give priority for color feature, this means gray feature were more compressed than color. Decompressor generate both features according to given priority and also two features were incorporated in the last layer of generator decoder network output block. Output block custom layer multiplication with the addition of instance normalization or batch normalization in the generator encoder and decompressor network while keeping auxiliary loss.

In the fourth experiment, similar to second and third experiments give priority for gray feature, this means color feature were more compressed than gray. Decompressor generate both features according to given priority and also two features were incorporated in the last layer of generator decoder network output block. Output block custom layer multiplication with the addition of instance normalization or batch normalization in the generator encoder and decompressor network while keeping auxiliary loss.

Table 5. 6 experiments class

Notation	Experiment	Dataset
Fidelity-controllable extreme image compression with GAN	Baseline	Kaggle Campinas of COCCO dataset for training and testing.
RPPIC-GAN-1 equal priority	Equal priority for both gray and color feature	
RPPIC-GAN-2 give priority to gray	Give priority to the gray feature more compress the color feature to lower bit rate	
RPPIC3-GAN give priority to color	Give priority to the color feature more compress the gray feature to lower bit rate	

CHAPTER SIX

6. RESULTS AND DISCUSSIONS

6.1. Overview of Results and Discussions

In preceding chapters, ideas are coined, methodologies, algorithms and models are proposed, designed and implemented. This chapter discus and analysis results, evaluate our models the various experiments take on and compares the results with figure, graphs, and tables. The algorithms including results gained from them and present comparative study done on baseline works and our work.

6.2. Result of the Study

6.2.1. Experimental Results

We carried out long training iterations utilizing the suggested networks, carefully including many parameters to increase the reliability of our research findings. In order to produce high-quality images within the training dataset, these studies targeted low bitrate image compression and included modifications in compression priority for extracted image features in grayscale and color. When compared to the baseline study, consistency was maintained in the dataset usage, software, and hardware configurations. We used the COCCO dataset for training and test used, to evaluate the effectiveness of our approach. Our outcomes were compared to modern low-bitrate image compression techniques, comparing our results with state-of-the-art image compression in low bitrate like Fidelity-Controllable Extreme Image Compression with GAN, the proposed relevant properties prioritize image compression using GAN and GAN for Extreme Learned Image Compression shows the best quantitative values and quantitative evaluation metrics.

6.2.2. Evaluation Metrics Results

Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) were used as two impartial metrics to thoroughly evaluate the model's performance. Table 6.1 provides a summary of the evaluation's findings and shows the generated images' quality as well as their compression at low bitrates. By measuring the similarity between the generated images and the ground truth reference images, the table also demonstrates the model's capacity to create images from low bitrates. We used the COCCO dataset, which has 91 items and 118,980 represent images for training and 40,000 for testing and original size of this dataset 640x640. Based on our resource for

train and test and size of images 25,096, 7,500 for training and testing by resize the size of images to 128x128. It is significant to note that greater PSNR and SSIM values indicate a tighter match between the generated and original images, indicating superior image quality and fidelity in the results.

6.2.2.1. Evaluation metrics all experiment

In Table 6.1, the evaluation of PSNR and SSIM metrics is used to determine how well the trained model reconstructs images by comparing how closely the rebuilt image resembles the original image. Increased PSNR and SSIM values are signs that the generated image and the original reference image are more closely related. These metrics essentially provide a measurable representation of the fidelity and quality attained throughout the image reconstruction process.

Table 6. 1 Quantitative results

Mask	BBP	Metrics	Evaluation result
Base line	0.033	PSNR	28.5
		SSIM	0.79
RPPIC1-GAN	0.0208	PSNR	29.77
		SSIM	**86.61**
RPPIC2-GAN	**0.018**	PSNR	29.45
		SSIM	85.55
RPPIC3-GAN	0.0200	PSNR	**30.50**
		SSIM	84.20

The bold value shows the best results in the experiments.

In the second experimental configuration (RPPIC1-GAN), SSIM shows a significant improvement (86.61) compared to the baseline (0.79). This enhancement could be attributed to the specific techniques or modifications incorporated in the RPPIC1-GAN model. These modifications might include better feature extraction, more accurate structural similarity measurement, or optimized training parameters that lead to improved structural similarity between the generated images and the ground truth. The improvements in SSIM across the different experimental configurations, particularly the substantial improvement in the second configuration (RPPIC1-GAN), indicate that

specific enhancements in the GAN models contribute to better preservation of structural similarity in the generated images when compared to the baseline configuration.

PSNR scores in the other experimental configurations (RPPIC1-GAN and RPPIC2-GAN) are also higher than the baseline (28.5), indicating that the GAN models (RPPIC1-GAN and RPPIC2-GAN) generally perform better in terms of noise reduction and image quality. However, the third configuration (RPPIC3-GAN) stands out with the highest PSNR, showcasing its superior performance in noise reduction compared to the other configurations. The improvements in PSNR across the different experimental configurations, particularly the significant improvement in the third configuration (RPPIC3-GAN), indicate that specific optimizations and advancements in the GAN models contribute to better noise reduction and image quality when compared to the baseline configuration.

6.3. Generated Image Results

6.3.1. Equal color and gray properties priority

Figure 6. 1 generated image of experiment

6.3.3. Prioritize gray properties as relevant

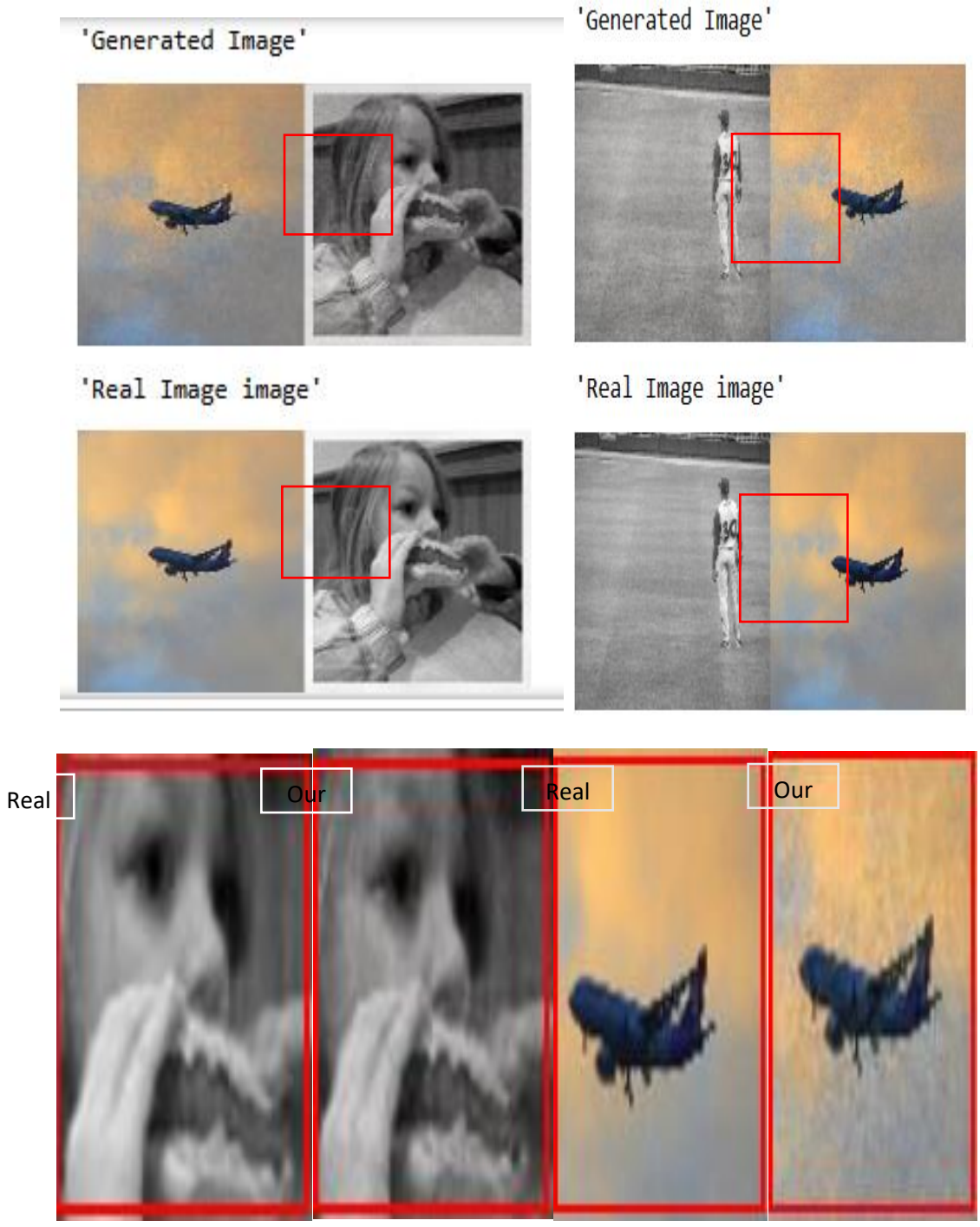

Figure 6. 2 generated image of experiment 3

2. Prioritize color properties as relevant

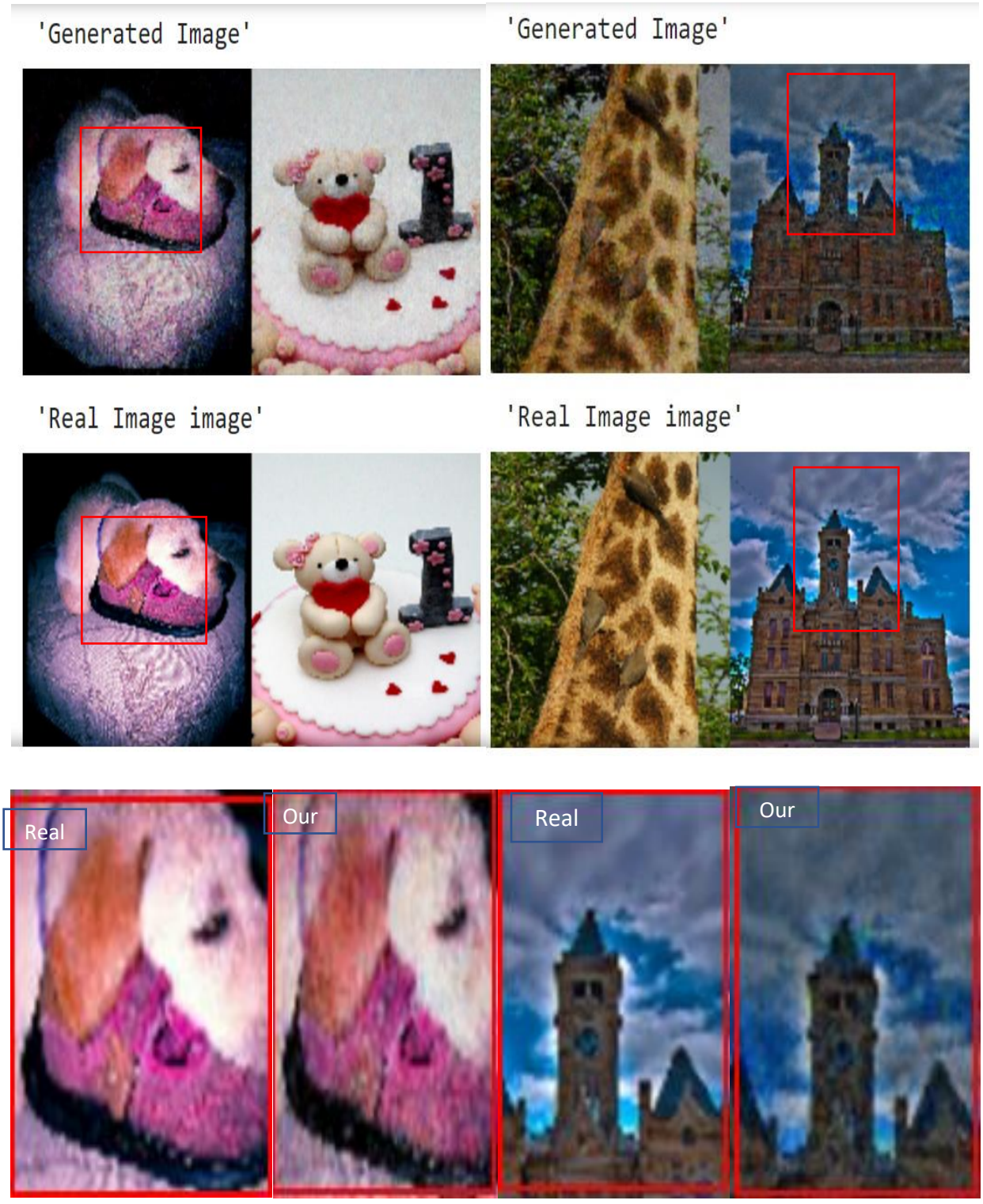

Figure 6. 3 generated image of experiment 3

6.4. Results Discussion

In this study, we have effectively demonstrated the efficacy of our proposed model in the selection of important features, which we refer to as relevant properties, and in assigning priorities during image compression using Generative Adversarial Networks (GAN). Incorporating a feature priority method based on Importance Area has allowed us to do this. Our research demonstrates that, across a range of quantitative indicators, our model outperforms other existing approaches, including those by with (Agustsson, Tschannen, Mentzer, Timofte Luc Van Gool, et al., 2019) Iwai et al., 2020). In particular, our model excels in improving images compression at low bitrates while keeping the generated images' quality and consistency in comparison to the original images. Our models have demonstrated a strong ability to reconstruct image details, maintain edges, and accurately represent information, all of which have helped to progress the field of image compression.

6.4.1. PSNR Results Discussion

The PSNR performance of three alternative compression and decompression methods— RPPIC1-GAN, RPPIC2-GAN, and RPPIC3-GAN —shows scores of 29.77, 29.45, and 30.50, respectively, when used the COCCCO dataset for training and testing. These results demonstrate the outstanding image reconstruction quality that these techniques were able to provide. Two other techniques—GAN for extreme learning image compression and Fidelity-controllable extreme image compression with GAN (the baseline technique)—were also assessed for context comparison. RPPIC3-GAN stands out among these techniques as the top performer by showcasing its extraordinary capacity to maintain image quality during high compression and considerably advancing image compression techniques.

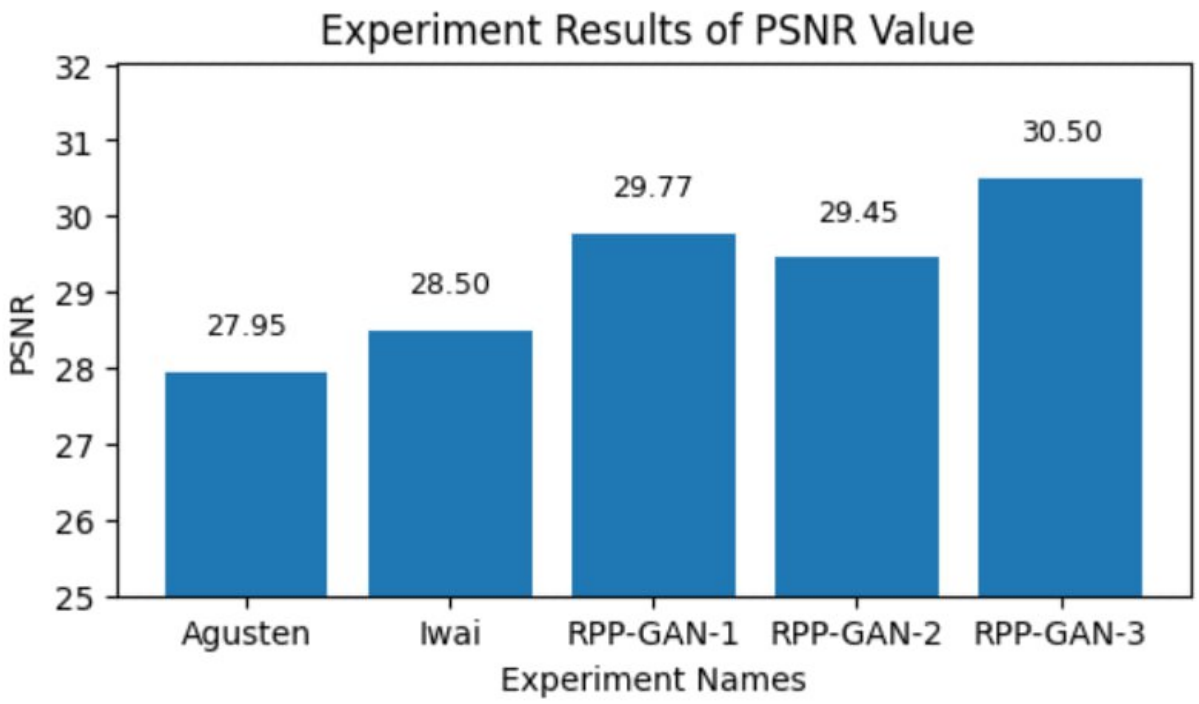

Figure 6.4 PSNR values for experiment

The experimental classes are displayed on the X-axis of the graph, and the peak Signal-to-Noise Ratio (PSNR) values are depicted on the Y-axis. The statistics in Figure 6.4 unquestionably demonstrate that the RPPIC3-GAN approach consistently beats the baseline strategy for a wide range of sample patterns. Notably, scores across all sampling patterns increase from **28.5 to 30.50**, demonstrating that it successfully completes the challenging task of decompressing images with extremely low bitrates. These findings show that the RPPIC3-GAN technique significantly outperforms the baseline method in situations requiring high-quality images reconstruction, especially when it comes to maintaining image quality and fidelity while subjected to excessive image compression.

6.4.2. Discussion on SSIM Results

The performance of our model's decompression results is depicted in Figure 6.5, using the Structural Similarity Index (SSIM) as the evaluation metric. RPPIC1-GAN, RPPIC2-GAN, and RPPIC3-GAN, the decompression techniques used in our model, are evaluated and contrasted. These methods are carefully compared to two competing strategies: Fidelity-controllable extreme image compression using GAN, which we use as our baseline approach, and GAN for extreme learnt image compression. This analysis offers important insights into how our suggested solutions

perform in terms of image fidelity and quality throughout the decompression stage and makes significant contributions to the field of image compression research.

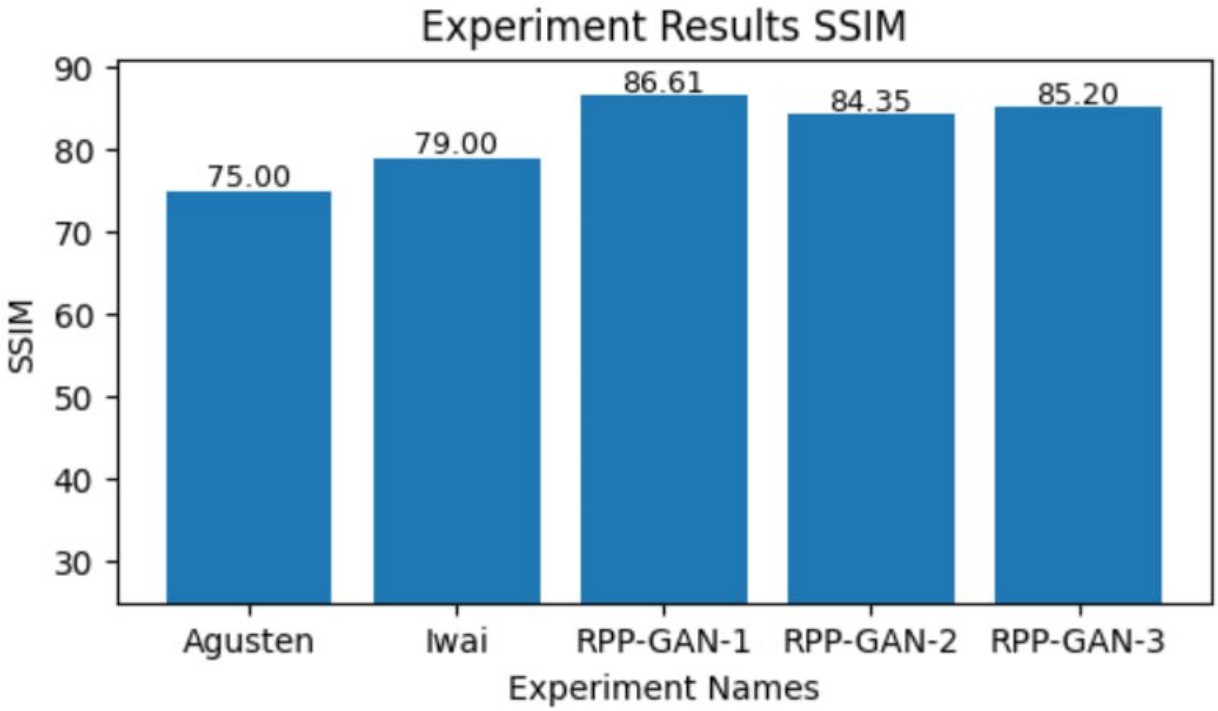

Figure 6. 5 SSIM values for experiment

The X-axis represents the experimental classes, and the Y-axis displays the Structural Similarity Index (SSIM) values. The statistics in Figure 6.5 clearly demonstrate that, regardless of the sample procedures employed, the RPPIC1-GAN strategy consistently outperforms all alternative methods. It achieves a great SSIM value ranging from **79 to 86.61** for extreme image compression with configurable quality. Importantly, these incredible results hold true for all sampling patterns even when applied to the challenging task of decompressing images with extremely low bitrates. The amazing abilities of the RPPIC1-GAN technique to produce both highly efficient image compression and precise control over image quality are highlighted by these results, making it a particularly attractive choice for scenarios demanding extreme image compression.

6.4.3. Discussion on Bit per Pixel results

The performance of our model's compression, as determined by the Bit per Pixel (BPP) metric, is shown in detail in Figure 6.6. The three compressor techniques being tested are RPPIC1-GAN, RPPIC2-GAN, and RPPIC3-GAN. We contrast these techniques with two alternative methods: GAN for extreme learning image compression and Fidelity-controllable extreme image

compression with GAN, which serves as our baseline technique, in order to contextualize and assess our findings. This research shows how our suggested compression approaches compare to existing ones in terms of efficiency, notably in terms of bits needed per pixel. These findings are important for the development of image compression since they show how successful our suggested methods are.

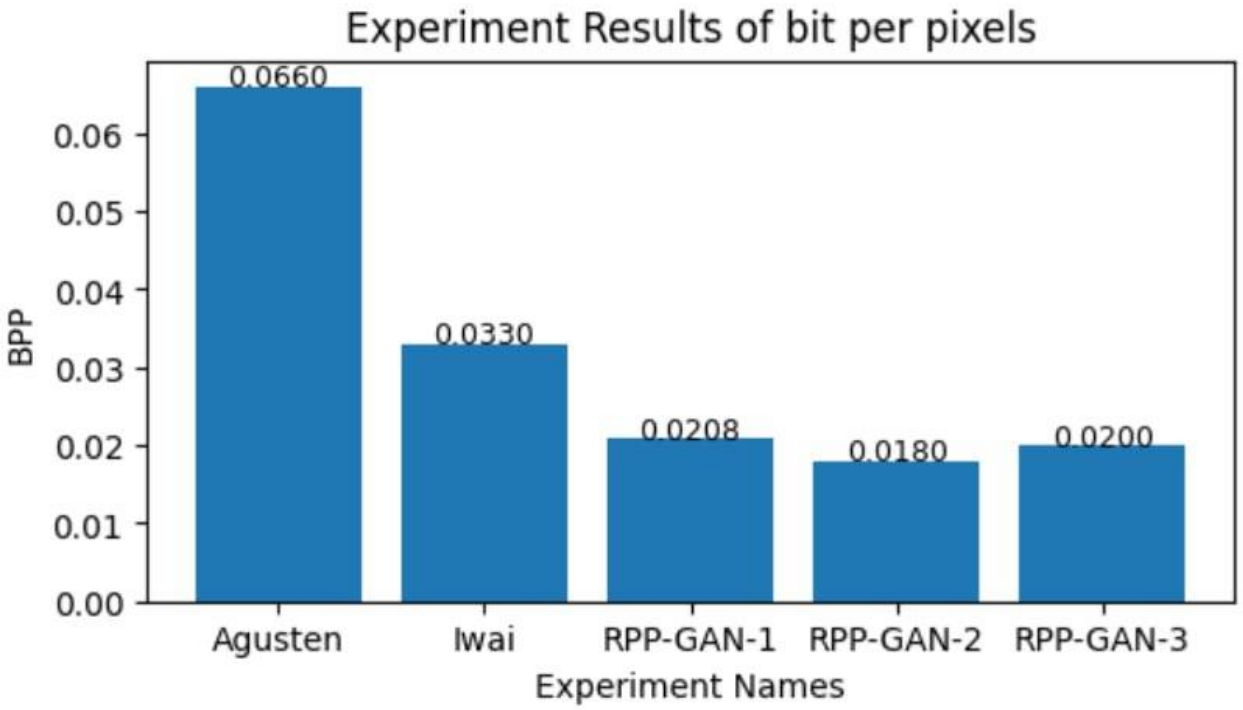

Figure 6. 6 BPP values for experiment

On the Y-axis of the graph are the Bit per Pixel (BPP) values, and on the X-axis are the experimental classes. The data shown in Figure 6.6 unambiguously shows that, across a wide range of sample patterns, the RPPIC2-GAN method performs better than all other methods. It delivers a remarkable BPP value for extreme image reduction with controlled fidelity using GAN, ranging from **0.0333 to 0.018**. Even when used to tackle the difficult task of decompressing images with extremely low bitrates, these astounding results stay true for all sampling patterns. These findings show how the RPPIC2-GAN technique excels at delivering extremely effective image compression while maintaining fidelity control, making it a particularly viable choice for situations requiring severe image compression.

6.5. Research Question and Answer Discussion

This study provided answers to all the research questions "**1.** What are the important features select and give priority as relevant properties in image compression through GAN implementation? **2.** What is the prioritized important features impact image compression and the resulting image quality when implemented with GANs?

Answer for Q1: In Section 2.1 of our study, we go into great detail about the essential qualities and advantages of image features. This section gives a comprehensive explanation of the key characteristics that make up images and emphasizes the benefits of learning more about them. Additionally, we explore the complexities of feature extraction in Section 3.4, clarifying how to derive and make use of the precise shades of these features.

We provide an architectural innovation to improve the compression and decompression of images within a generator framework in our suggested approach, as described in Section 4.2 of our study. This novel strategy makes use of an internal compressor layer that dynamically merges down-sample blocks according to predetermined priority criteria. The decompressor component is also thoroughly made to reconstruct each image feature from the output of the compressed model. The generated final image is then produced by processing each of these distinct features using a customized custom multiplication layer.

Answer for Q2: In this study, the compressor and decompressor networks both included a feature prioritizing algorithm. This approach is important for considerably minimize image size while also improving the quality of the generated images at lower bitrates, in line with user needs. The findings of a thorough evaluation based on the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) results clearly demonstrated the high quality of these generated images. These assessments were carried out as part of our experimental investigation, and the thorough results and findings are shown in Tables 6.1 and section of the 6.2, 6.3 and 6.4 demonstrating the effectiveness of our strategy for low bitrate image compression

CHAPTER SEVEN

7. CONCLUSION AND FUTURE WORKS

7.1. Conclusion

In this study, we introduced a novel solution to the image compression problem based on the deep learning-based RPPIC1-GAN (Relevant Properties Prioritized Image Compression Generative Adversarial Network) model. Our research intends to enhance the performance of generative adversarial networks (GANs), particularly for low bitrate image compression. The major objective was to efficiently compress images at low bitrates by identifying and emphasizing key visual components while underlining less significant ones. In contrast to past methods like Fidelity-Controllable Extreme Image Compression, which struggled with low bitrates and failed to improve image quality, our study offered a revolutionary solution leveraging a Generative Adversarial Network (GAN) architecture. The generator in this architecture consists of one compressor and two decompressors. One compressor and two decompressors make up the generator in his architecture. The compressor is a specifically created custom residual network in down sampling and skip blocks in the compressor for feature prioritization. In order to ensure the reconstruction of all necessary image features, the compressor network generates features for decompression. To improve reconstruction for very low bitrate image compression, we used an adversarial setup throughout the training phase with a standard discriminator WGAN (Wasserstein GAN) loss functions.

Several experiments were conducted simultaneously to make comparisons. The first experiment is RPPIC1-GAN which compress equal priority of feature compression in compressor, in this experiment reduce bit per pixel image size in to 0.0208 and generated quality of images measured by PSNR and SSIM values 29.77 and 86.61. The second experiment is RPPIC2-GAN which compress give priority to the gray feature, in this experiment more compress the color than gray feature in compressor network, in this experiment reduce bit per pixel image size in to 0.018 and generated quality of images measured by PSNR and SSIM values 29.45 and 84.35. The second experiment is RPPIC3-GAN which compress give priority to the color feature, in this experiment more compress the gray feature than color in compressor network, in this experiment reduce bit per pixel image size in to 0.020 and generated quality of images measured by PSNR and SSIM values 30.5 and 85.2. Finally, this proposed learned image compression in low bitrate relevant

properties prioritize based on GAN, has tried to contribute further improvement on the generative architecture to produce better results in the compression and generating quality images, this works reliability to compare the low bitrate image compression.

7.2. Our Work Future Works

In our future work, we wish to investigate the potential for expanding the architecture's our work capabilities with more image features in order to better understand it and support its efficacy. By supporting various compression ratios, for instance, we may more effectively reduce the size of images while keeping quality images to improved decompression networks. This will expand the use of our GAN. Although our study initially focused on training with just two feature color and gray, we notice the potential benefits of using a more than two features according to importance. Additionally, we aim to change to unsupervised training methods, particularly in situations where access to fully sampled data is constrained, and modify the input configuration of the compressor network to flexibly accommodate various quantities of features with their importance for image compression. Our research is also expanding in the following directions: investigating more than two feature selection for image compression properties. These collective efforts will greatly increase the capability and efficacy of our image compression technique.

* * *

Special Acknowledgment

This research work was funded by grant number ASTU/SM-R/812/23 from Adama Science and Technology University, **Adama, Ethiopia**.

* * *

REFERENCES

Abdel-Salam Nasr, M., Mohammed F. AlRahmawy, and A. S. Tolba. 2017. "Multi-Scale Structural Similarity Index for Motion Detection." *Journal of King Saud University - Computer and Information Sciences* 29(3):399–409. doi: 10.1016/J.JKSUCI.2016.02.004.

Agustsson, Eirikur, Michael Tschannen, Fabian Mentzer, Radu Timofte Luc, and Van Gool. n.d.-a. *Extreme Learned Image Compression with GANs*.

Agustsson, Eirikur, Michael Tschannen, Fabian Mentzer, Radu Timofte Luc, and Van Gool. n.d.-b. *Extreme Learned Image Compression with GANs*.

Agustsson, Eirikur, Michael Tschannen, Fabian Mentzer, Radu Timofte Luc Van Gool, and Eth Zürich. 2019. "Generative Adversarial Networks for Extreme Learned Image Compression." 221–31.

Agustsson, Eirikur, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. 2019. "Generative Adversarial Networks for Extreme Learned Image Compression." *Proceedings of the IEEE International Conference on Computer Vision* 2019-Octob:221–31. doi: 10.1109/ICCV.2019.00031.

Akutsu, Hiroaki, Akifumi Suzuki, Zhisheng Zhong, and Kiyoharu Aizawa. 2020. "Ultra Low Bitrate Learned Image Compression by Selective Detail Decoding." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2020-June:524–28. doi: 10.1109/CVPRW50498.2020.00067.

Anon. n.d.-a. "Advantages Of Image Compression - 3267 Words | Internet Public Library." Retrieved December 19, 2022 (https://www.ipl.org/essay/Advantages-Of-Image-Compression-FCWAM9L4RU).

Anon. n.d.-b. "GitHub - MohamedBakrAli/Kodak-Lossless-True-Color-Image-Suite: Dataset." Retrieved December 18, 2022 (https://github.com/MohamedBakrAli/Kodak-Lossless-True-Color-Image-Suite).

Anon. n.d.-c. "True Color Kodak Images." Retrieved December 10, 2022 (https://r0k.us/graphics/kodak/).

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. 2017. "Wasserstein GAN."

Ballé, Johannes, Valero Laparra, and Eero P. Simoncelli. 2016. "End-to-End Optimized Image Compression." *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. doi: 10.48550/arxiv.1611.01704.

Cheng, Zhengxue, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2020. "Learned Image Compression With Discretized Gaussian Mixture Likelihoods and Attention Modules." 7939–48.

Chen, Huan, Yue Hsien Wang, and Chun Hung Fan. 2021. "A Convolutional Autoencoder-Based Approach with Batch Normalization for Energy Disaggregation." *Journal of Supercomputing* 77(3):2961–78. doi: 10.1007/s11227-020-03375-y.

Chen, Tong, Haojie Liu, Zhan Ma, Qiu Shen, Xun Cao, and Yao Wang. 2019. “Neural Image Compression via Non-Local Attention Optimization and Improved Context Modeling.” doi: 10.48550/arxiv.1910.06244.

Chen, Tong, Haojie Liu, Zhan Ma, Qiu Shen, Xun Cao, and Yao Wang. 2021. “End-to-End Learnt Image Compression via Non-Local Attention Optimization and Improved Context Modeling.” *IEEE Transactions on Image Processing* 30:3179–91. doi: 10.1109/TIP.2021.3058615.

Das, Kalyan, Jiming Jiang, and J. N. K. Rao. 2004. *MEAN SQUARED ERROR OF EMPIRICAL PREDICTOR*. Vol. 32.

Van Den, Aäron, Oord Google Deepmind, Nal Kalchbrenner, Google Deepmind, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016. “Conditional Image Generation with PixelCNN Decoders.” *Advances in Neural Information Processing Systems* 29.

Dua, Yaman, Ravi Shankar Singh, Kshitij Parwani, Smit Lunagariya, and Vinod Kumar. 2021. “Convolution Neural Network Based Lossy Compression of Hyperspectral Images.” *Signal Processing: Image Communication* 95. doi: 10.1016/j.image.2021.116255.

Du, Hanwen, Huanhuan Yuan, Zhen Huang, Pengpeng Zhao, and Xiaofang Zhou. 2023. “Sequential Recommendation with Diffusion Models.” 17. doi: 10.1145/nnnnnnn.nnnnnnn.

Emami, Hajar, Majid Moradi Aliabadi, Ming Dong, and Ratna Babu Chinnam. 2021a. “SPA-GAN: Spatial Attention GAN for Image-to-Image Translation.” *IEEE Transactions on Multimedia* 23:391–401. doi: 10.1109/TMM.2020.2975961.

Emami, Hajar, Majid Moradi Aliabadi, Ming Dong, and Ratna Babu Chinnam. 2021b. “SPA-GAN: Spatial Attention GAN for Image-to-Image Translation.” *IEEE Transactions on Multimedia* 23:391–401. doi: 10.1109/TMM.2020.2975961.

Galteri, Leonardo, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. n.d. “Deep Generative Adversarial Compression Artifact Removal.”

Glover-Kapfer, Paul, Carolina A. Soto-Navarro, Oliver R. Wearn, Correspondence A. Carolina Soto-Navarro, and Un Environment. 2019. “Camera-Trapping Version 3.0: Current Constraints and Future Priorities for Development.” *Remote Sensing in Ecology and Conservation* 5(3):209–23. doi: 10.1002/RSE2.106.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. “Generative Adversarial Networks.” *Communications of the ACM* 63(11):139–44. doi: 10.1145/3422622.

Guo, Hui, Shu Hu, Xin Wang, Ming Ching Chang, and Siwei Lyu. 2022. “Robust Attentive Deep Neural Network for Detecting GAN-Generated Faces.” *IEEE Access* 10:32574–83. doi: 10.1109/ACCESS.2022.3157297.

Hassan, Esraa, Mahmoud Y. Shams, Noha A. Hikal, and Samir Elmougy. 2023. “The Effect of Choosing Optimizer Algorithms to Improve Computer Vision Tasks: A Comparative Study.” *Multimedia Tools and Applications* 82(11):16591–633. doi: 10.1007/s11042-022-13820-0.

Huang, Junzhou, Shaoting Zhang, and Dimitris Metaxas. 2010. “Efficient MR Image Reconstruction for Compressed MR Imaging.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6361 LNCS(PART 1):135–42. doi: 10.1007/978-3-642-15705-9_17/COVER.

Hussain, A. J., Ali Al-Fayadh, and Naeem Radi. 2018. “Image Compression Techniques: A Survey in Lossless and Lossy Algorithms.” *Neurocomputing* 300:44–69. doi: 10.1016/J.NEUCOM.2018.02.094.

Hu, Yueyu, Wenhan Yang, Zhan Ma, and Jiaying Liu. 2022. “Learning End-to-End Lossy Image Compression: A Benchmark.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44(8):4194–4211. doi: 10.1109/TPAMI.2021.3065339.

Im, Daniel Jiwoong, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. 2016. “Generating Images with Recurrent Adversarial Networks.”

Iwai, Shoma, Tomo Miyazaki, Yoshihiro Sugaya, and Shinichiro Omachi. 2020. “Fidelity-Controllable Extreme Image Compression with Generative Adversarial Networks.” *Proceedings - International Conference on Pattern Recognition* 8235–42. doi: 10.1109/ICPR48806.2021.9412185.

Jamil, Sonain, Md. Jalil Piran, and MuhibUrRahman. 2022. “Learning-Driven Lossy Image Compression; A Comprehensive Survey.” 1–14.

Jay, Florence, Jean-Pierre Renou, Olivier Voinnet, and Lionel Navarro. 2017. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks Jun-Yan.” *Proceedings of the IEEE International Conference on Computer Vision* 183–202.

Kingma, Diederik P., and Jimmy Ba. 2014. “Adam: A Method for Stochastic Optimization.”

Knyaz, Vladimir A., Vladimir V. Kniaz, and Fabio Remondino. 2019a. “Image-to-Voxel Model Translation with Conditional Adversarial Networks.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11129 LNCS:601–18. doi: 10.1007/978-3-030-11009-3_37.

Knyaz, Vladimir A., Vladimir V. Kniaz, and Fabio Remondino. 2019b. “Image-to-Voxel Model Translation with Conditional Adversarial Networks.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11129 LNCS:601–18. doi: 10.1007/978-3-030-11009-3_37.

Lee, Jooyoung, Seunghyun Cho, and Munchurl Kim. 2019. “An End-to-End Joint Learning Scheme of Image Compression and Quality Enhancement with Improved Entropy Minimization.” doi: 10.48550/arxiv.1912.12817.

Li, Jin, and Zilong Liu. 2019. “Multispectral Transforms Using Convolution Neural Networks for Remote Sensing Multispectral Image Compression.” *Remote Sensing 2019, Vol. 11, Page 759* 11(7):759. doi: 10.3390/RS11070759.

Lin, Tsung Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. “Microsoft COCO: Common Objects in Context.” *Lecture Notes in*

Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8693 LNCS(PART 5):740–55. doi: 10.1007/978-3-319-10602-1_48/COVER.

Lu, Ming, Student Member, Fangdong Chen, Shiliang Pu, Zhan Ma, and Senior Member. n.d. "High-Efficiency Lossy Image Coding Through Adaptive Neighborhood Information Aggregation." (1).

Mentzer, Fabian, Eth Zürich, George Toderici, Michael Tschannen, Eirikur Agustsson, and Google Research. 2020. "High-Fidelity Generative Image Compression." *Advances in Neural Information Processing Systems* 33:11913–24.

Minnen, David, and Saurabh Singh. 2020. "Channel-Wise Autoregressive Entropy Models for Learned Image Compression." *Proceedings - International Conference on Image Processing, ICIP* 2020-October:3339–43. doi: 10.1109/ICIP40778.2020.9190935.

Nair, Suraj, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. 2019. "Causal Induction from Visual Observations for Goal Directed Tasks."

Preti, Michele, François Verheggen, and Sergio Angeli. 2021. "Insect Pest Monitoring with Camera-Equipped Traps: Strengths and Limitations." *Journal of Pest Science* 94(2):203–17. doi: 10.1007/S10340-020-01309-4/TABLES/2.

Quan, Tran Minh, Thanh Nguyen-Duc, and Won Ki Jeong. 2018. "Compressed Sensing MRI Reconstruction Using a Generative Adversarial Network With a Cyclic Loss." *IEEE Transactions on Medical Imaging* 37(6):1488–97. doi: 10.1109/TMI.2018.2820120.

Sara, Umme, Morium Akter, and Mohammad Shorif Uddin. 2019a. "Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study." *Journal of Computer and Communications* 07(03):8–18. doi: 10.4236/jcc.2019.73002.

Sara, Umme, Morium Akter, and Mohammad Shorif Uddin. 2019b. "Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study." *Journal of Computer and Communications* 07(03):8–18. doi: 10.4236/jcc.2019.73002.

Sento, Adna. 2017. "Image Compression with Auto-Encoder Algorithm Using Deep Neural Network (DNN)." *2016 Management and Innovation Technology International Conference, MITiCON 2016* MIT99–103. doi: 10.1109/MITICON.2016.8025238.

Skodras, Athanassios, Charilaos Christopoulos, and Touradj Ebrahimi. 2001. "The JPEG 2000 Still Image Compression Standard." *IEEE Signal Processing Magazine* 18(5):36–58. doi: 10.1109/79.952804.

Sun, Hao, Xiangtao Zheng, Xiaoqiang Lu, and Siyuan Wu. 2020. "Spectral-Spatial Attention Network for Hyperspectral Image Classification." *IEEE Transactions on Geoscience and Remote Sensing* 58(5):3232–45. doi: 10.1109/TGRS.2019.2951160.

Torfason, Robert, Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018a. "Towards Image Understanding from Deep Compression without Decoding." *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Torfason, Robert, Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018b. "Towards Image Understanding from Deep Compression without Decoding."

Tschannen, Michael, Eth Zürich, Eirikur Agustsson, Mario Lucic, and Google Brain. 2018. "Deep Generative Models for Distribution-Preserving Lossy Compression." *Advances in Neural Information Processing Systems* 31.

Vafaeikia, Partoo, Khashayar Namdar, and Farzad Khalvati. 2020. "A Brief Review of Deep Multi-Task Learning and Auxiliary Task Learning."

Vallecorsa, Sofia, Federico Carminati, and Gulrukh Khattak. 2019. "3D Convolutional GAN for Fast Simulation." *EPJ Web of Conferences* 214:02010. doi: 10.1051/epjconf/201921402010.

Viroli, Cinzia, and Geoffrey J. McLachlan. 2019. "Deep Gaussian Mixture Models." *Statistics and Computing* 29(1):43–51. doi: 10.1007/S11222-017-9793-Z/TABLES/2.

Wallace, Gregory K. 1991. "The JPEG Still Picture Compression Standard." *Communications of the ACM* 34(4):30–44. doi: 10.1145/103085.103089.

Wang, Xintao, Ke Yu, Chao Dong, Xiaoou Tang, and Chen Change Loy. 2019. "Deep Network Interpolation for Continuous Imagery Effect Transition." 1692–1701.

Wu, Lirong, Kejie Huang, and Haibin Shen. 2020. "A GAN-Based Tunable Image Compression System." *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020* 2323–31. doi: 10.1109/WACV45572.2020.9093387.

Zeegers, Mathé T., Daniël M. Pelt, Tristan van Leeuwen, Robert van Liere, and Kees Joost Batenburg. 2020. "Task-Driven Learned Hyperspectral Data Reduction Using End-to-End Supervised Deep Learning." *Journal of Imaging 2020, Vol. 6, Page 132* 6(12):132. doi: 10.3390/JIMAGING6120132.

Zhang, Hai Miao, and Bin Dong. 2020. "A Review on Deep Learning in Medical Image Reconstruction." *Journal of the Operations Research Society of China* 8(2):311–40. doi: 10.1007/S40305-019-00287-4/FIGURES/6.

Zhang, Xi, Shanghai Jiao, and Xiaolin Wu. n.d. "Attention-Guided Image Compression by Deep Reconstruction of Compressive Sensed Saliency Skeleton."