Image Restoration using GAN

Submitted 6 November 2023, in partial fulfillment of
the conditions for the award of the degree **BSc in Internet of Things and Robotics Engineering .**

Jul Jalal Al Mamur Sayor, Nishat Tasnim Shishir, Fahim Shahriar Prottoy
1901029, 1901030, 1901041

Supervised by Mahir Mahbub

Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh.

Abstract

In this paper, we address the challenge of image restoration, especially in cases of extensive damage or pixelation. Traditional methods often fall short in such scenarios, prompting the exploration of Deep Neural Networks (DNNs) for improved results. We introduce a novel approach that leverages a Generative Adversarial Network (GAN) for image restoration. Our system consists of an image generation network responsible for restoring damaged images and a discriminator network that assesses the authenticity of the restorations. Additionally, we provide a user-friendly web interface for real-time image restoration. This project primarily highlights the effectiveness of our image restoration network, emphasizing its prowess in scenarios involving pixelation and widespread damage.

Keywords: Deep Neural Networks (DNNs), Image Restoration, Generative Adversarial Network (GAN),Image Generation Network, Damage Reconstruction.

Acknowledgements

We extend our sincere appreciation to the individuals whose contributions have played a pivotal role in the successful completion of this project. Foremost, we express our profound gratitude to our esteemed project supervisor, Mahir Mahbub sir, whose unwavering support, expert guidance, and profound insights have been indispensable throughout the project's lifecycle. His mentorship has not only steered our project in the right direction but has also significantly enriched the depth and quality of this paper. We are deeply indebted to our classmates who have generously offered their collaboration and engaged in fruitful discussions, fostering an environment of intellectual exchange that has critically shaped the project's outcomes. The collective contributions of all individuals mentioned above have collectively enabled the fruition of this work, and we remain profoundly grateful for the opportunities and resources that have facilitated our project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One enduring difficulty associated with the widespread use of digital imaging across multiple areas is the restoration of damaged photographs. From pixelation to extensive damage, the need to recover visual content with precision and efficiency is a fundamental problem in the realm of digital image processing. While algorithms for image reconstruction have made substantial strides, they often fall short of the precision and nuance achieved through manual restoration. The emergence of Deep Neural Networks (DNNs) has opened up a new frontier in image restoration, ushering in a wave of innovative approaches aimed at leveraging the power of machine learning to address this challenge. This project embarks on a journey to explore the potential of DNNs, particularly the use of Generative Adversarial Networks (GANs), to transform the landscape of image restoration. Our overarching intention is to develop an automated system capable of effectively restoring damaged images.

1.1 Motivation

The motivation behind this project is rooted in the enduring need to enhance the quality of digital images that have been subjected to various forms of damage or degradation. In the digital age, images play a pivotal role in numerous sectors, including medical imaging, entertainment, surveillance, and more. These images often undergo challenges, such as pixelation, noise, or extensive damage, due to data transmission errors, compression, or environmental factors. Restoring these damaged images is not merely a technical challenge; it carries significant implications for data integrity, visual aesthetics, and decision-making in critical domains. The primary motivation for embarking on this project stems from several key considerations:

1. Quality Enhancement
2. Time and Cost Efficiency
3. Broad Applicability
4. User Accessibility
5. Technological Advancement

1.2 Aims and Objectives

1.2.1 Aims

The aim of this project is to develop an innovative image restoration system leveraging Generative Adversarial Networks (GANs) to efficiently and effectively restore damaged digital images, with a particular focus on scenarios where traditional interpolation methods prove inadequate.

1.2.2 Objectives

1. Design and Implement an Image Restoration Network: Develop an image generation network that is capable of restoring damaged images with a high degree of accuracy and detail. This network will serve as the core component of the image restoration system.
2. Create an Authenticity Evaluator (Discriminator): Design and implement a discriminator network that can evaluate the authenticity of the generated image restorations. This discriminator will assist in ensuring the quality and realism of the restored images.
3. Optimize GAN Model Training: Train the GAN model using a carefully curated dataset of damaged and undamaged images. Optimize the training process to achieve high-quality image restorations and minimize artifacts or inconsistencies.

4. Develop a User-Friendly Web Interface: Create a web interface using flask, which will allow users to upload damaged images and observe the real-time image restoration process. The interface should be intuitive and accessible to users with varying levels of technical expertise.
5. Performance Assessment: Evaluate the effectiveness of the image restoration system by assessing its performance on a diverse range of damaged images, including those with extensive and intricate damage. Measure restoration quality in terms of visual fidelity and accuracy.

1.3 Description of the work

This project is intended to deliver an innovative image restoration system designed to effectively restore damaged digital images. The core objective is to harness the capabilities of Generative Adversarial Networks (GANs) to address scenarios where traditional interpolation methods fall short in restoring image quality. The system is meant to function as an automated and user-friendly tool for image restoration, offering a practical solution for enhancing the visual quality of images.

1.3.1 Project functionality

1. Image Restoration
2. Generative Adversarial Networks (GANs)
3. Dataset and Training
4. Web Interface
5. Quality Assessment

Chapter 2

Background and Related Work

In 2014, Ian Goodfellow et al. Proposed a Generative Adversarial Network (GAN)[1], and provided a framework and theoretical convergence analysis of a Generative Adversarial Network (GAN). The core idea of generative adversarial network is to learn the corresponding probability distribution from the training samples, and obtain more generated samples based on the probability distribution. Generative Adversarial Networks are an example of advanced deep learning models with two major components, a generator and a discriminator. Generative Adversarial Nets were proposed in 'I. Goodfellow, "NASA/ADS", Ui.adsabs.harvard.edu, 2019. [Online]. Available: https://ui.adsabs.harvard.edu/. [Accessed: 01-Dec-2019]' wherein generator and discriminator models were separately trained, wherein generator estimates the probability distribution of the data and discriminator aims to predict if the test input came from training data. Typically, the generator is given a random noise input of n-dimensions, sampled from a predefined latent space (e.g. multi variate normal distribution). Thereafter, discriminator evaluates the candidates synthesized by the generative network or generator. To allow generator produce better images, backpropagation is applied to both the neural networks, while the discriminator becomes better by flagging or identifying the synthesized image candidates. The generator is typically a deconvolutional neural network, and the discriminator is a convolutional neural network[3]. The network framework[1] of GAN is shown in 2.1. First, a dimensional noise vector is sampled from the latent space and input to a generator network. The generator converts the noise vector into an image, and the generated image is sent to a discriminator network for classification. The discriminating object of the discriminator is constantly switched between the real data set and the image generated by the generator, and the discriminator parameters are updated according to the respective discriminating errors. All GAN architectures basically follow this design idea. Recently, deep learning has been applied to many image restoration tasks[5]. Since the appearance of photographs technologies, taking photographs becomes an effective way to keep people's daily life into memory. In the age of economic underdevelopment, people generally leave a few photographs keeping precious memories for the future. However, with time passing, many factors spoil photographs, such as creases, spots, water, light, or human-made factors. Consequently, spoiled photographs restoration has become a significant thing and has been widely concerned by academia and society. One usually uses image processing software, such as Photoshop (PS) and Metuxiu, to repair old and damaged photographs, at the cost of a long time and a lot of human resources. In recent years, artificial intelligence (AI) and deep learning (DL) have been widely applied in the field of image processing [4]. In our project we use GAN for image restoration process.

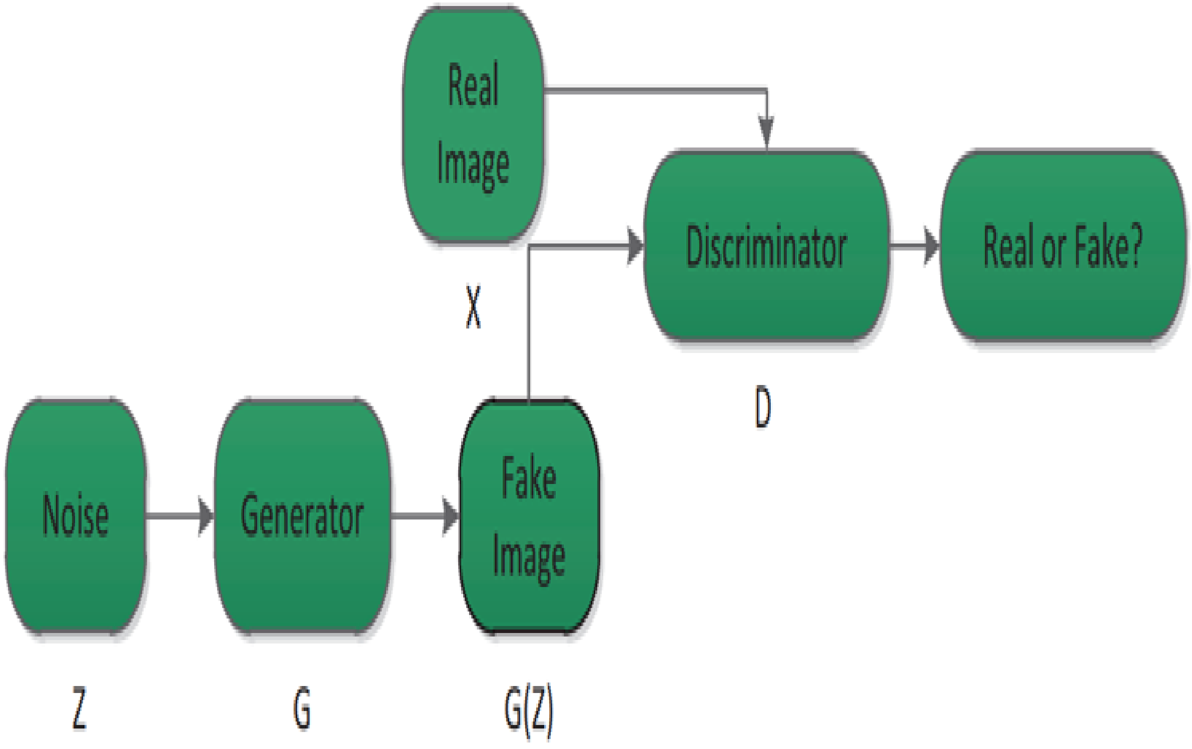


Figure 2.1: GAN Network Framework

Chapter 3

Design

The project system utilizes a GAN-based architecture consisting of a generator and a discriminator. The generator is responsible for generating high-quality images from low-quality inputs, while the discriminator evaluates the authenticity of the generated images compared to the ground truth. The design employs upsampling and downsampling for the generator, facilitating the preservation of intricate details during the image restoration process. Additionally, the discriminator utilizes a convolutional neural network (CNN) to distinguish between real and generated images, ensuring the production of visually appealing and realistic results.To address the problem of image restoration, the design incorporates a combination of adversarial and perceptual loss functions. The adversarial loss encourages the generator to produce images that are indistinguishable from the ground truth, while the perceptual loss ensures that the generated images preserve the structural and semantic information of the original content. The use of a weighted combination of these loss functions, along with the incorporation of feature matching techniques, enables the model to effectively restore images while maintaining their authenticity and visual coherence.The design integrates batch normalization and LeakyReLU activation functions within the network architecture to stabilize the training process and prevent issues such as vanishing gradients. The inclusion of skip connections facilitates the flow of information between the encoder and decoder, allowing the network to capture both global and local features of the input images. Moreover, the utilization of Adam optimizer with an adaptive learning rate enhances the convergence speed and overall performance of the model during the training phase. Users can easily interact with the image restoration system through a user-friendly web interface. Users can upload pixelated images, and the system returns restored versions of those images. The project's web architecture is built using the Flask framework.

The project's coding algorithm is given below:

```
Step 1: Import necessary libraries
Step 2: Set image size, variables, and file paths
Step 3: Define a function for sorting image files
Step 4: Load and preprocess color and grayscale images
Step 5: Create TensorFlow datasets for training and testing data
Step 6: Define and initialize Generator and Discriminator models
Step 7: Set up loss functions and optimizers
Step 8: Create a training loop
for epoch in range(epochs):
Step 9: Inside the training loop
for batch in training dataset:
a. Forward and backward passes through Generator and Discriminator
b. Calculate and update gradients
c. Apply gradients to optimize Generator and Discriminator Periodically display sample images
Step 10: After training, save Generator and Discriminator models
Step 11: End
```

Chapter 4

Implementation

4.1 Dataset

In this project we are using a dataset named "humanface8000" from Kaggle. The dataset contains 16.3k files. Where in the landscape images there is two directories named "color" and "gray". The color directory contains 8164 files and the gray directory contains 8164 files. It can be used for old image restoration using deep learning. This dataset is curated automatically through the use of filter to make noise/scratches in the image using a function and also used an overlay image to map those scratches.

4.2 Model Formation

In this project we have to first loading and preprocess color and gray images from the dataset. Then we have to define the generator and discriminator model. A standard Convolutional Neural Network (CNN) architecture is used for both the generator and discriminator models. The CNN architecture consists of convolutional layers, batch normalization layers, and activation functions (LeakyReLU) to process and transform the input data. These layers are organized into downsample and upsample blocks in the generator and downsample blocks in the discriminator.

4.2.1 Generator Model

Downsample blocks: These are comprised of Conv2D layers followed by batch normalization and LeakyReLU activation functions.
Upsample blocks: These are also comprised of Conv2DTranspose layers followed by batch normalization and ReLU activation functions.
Hidden Layer in the generator model:
The downsample block includes seven layers:
downsample(64, 4), downsample(128, 4), downsample(256, 4), downsample(512, 4), downsample(512, 4), downsample(512, 4), downsample(512, 4).
The upsample block includes six layers:
upsample(512, 4), upsample(512, 4), upsample(512, 4), upsample(512, 4), upsample(256, 4), upsample(128, 4).
There is a final convolutional layer also.
So, in total, there are 7 + 6 + 1 = 14 hidden layers in the generator model.

4.2.2 Discriminator Model

Downsample blocks: These are made up of Conv2D layers followed by batch normalization and LeakyReLU activation functions.
The downsample block includes three layers: downsample(64, 4), downsample(128, 4), downsample(256, 4).
After the downsample block, there are a few additional layers, including convolutional and batch normalization layers.
So, in total, there are *3 + additional layers in the discriminator model.
The exact number of additional layers in the discriminator model depends on the specific structure of downsample function and the layers used after the downsample block.

The choice of convolutional layers and the use of batch normalization and activation functions make this architecture a standard CNN used for image-to-image translation tasks, as is common in GANs.

4.2.3 Downsampling

The downsample function is used to reduce the spatial dimensions of the input while increasing the number of channels (filters) through convolutional layers.the ”Generator” function includes downsample layers.

4.2.4 Upsampling

The upsample function is used to increase the spatial dimensions of the input while decreasing the number of channels (filters).the ”Generator” function includes upsample layers.

4.2.5 Batch Normalization

In the discriminator model we use batch normalization. BatchNorm is used within the generator model to normalize the activations in various layers. It contributes to the stability, faster training, and improved performance of the generator when learning to colorize grayscale images. Without BatchNorm, training deep neural networks, especially GANs, can become more challenging and less effective. Here we provide two figures for understanding the effect of batch normalization[2]. The figures are not for our project's data.

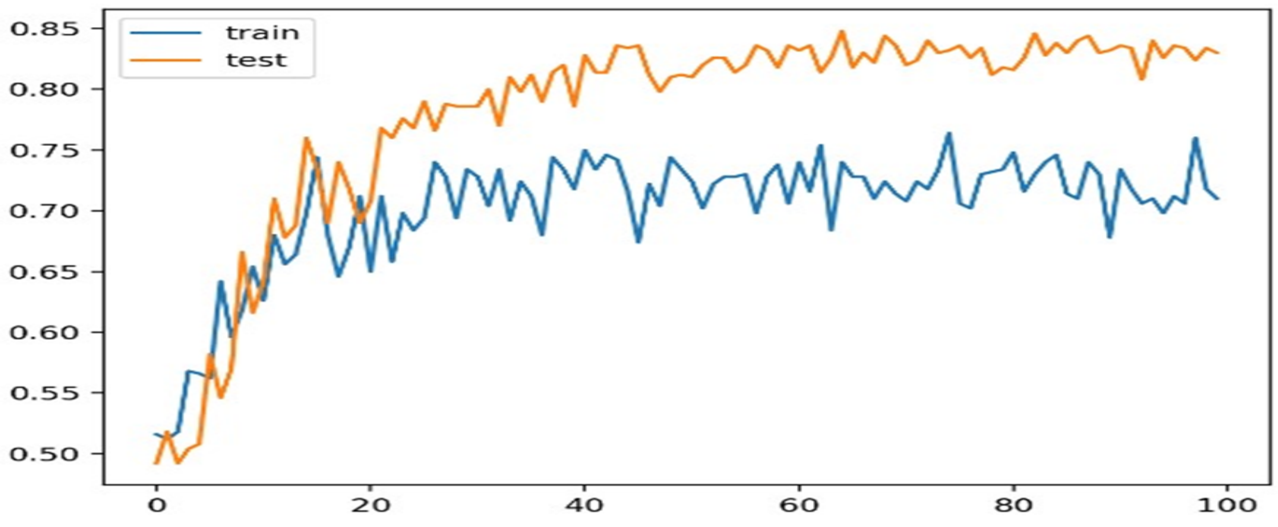


Figure 4.1: Before Batch Normalization.

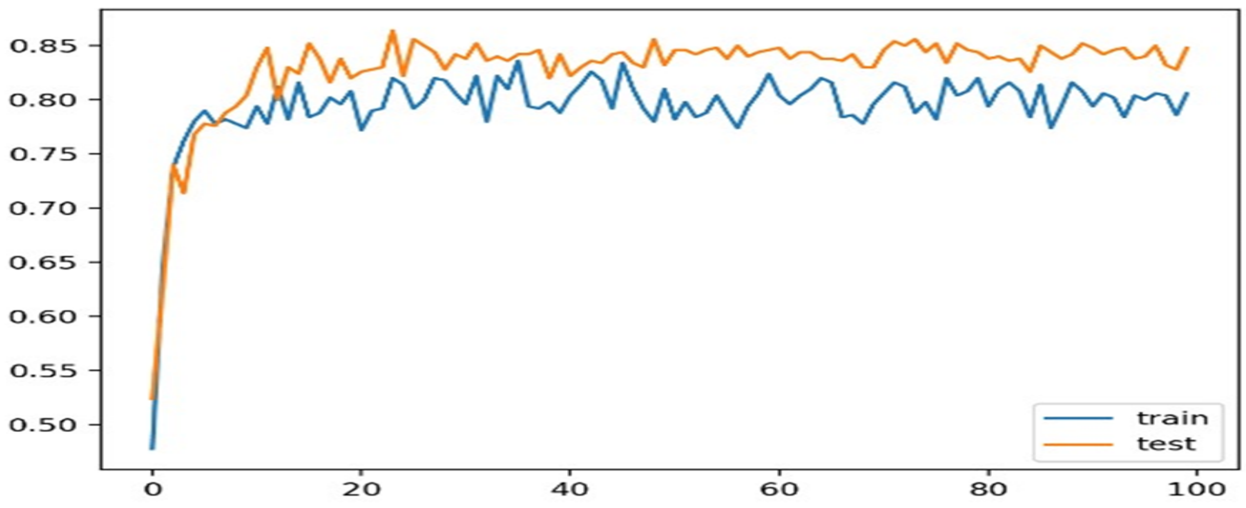


Figure 4.2: After Batch Normalization.

4.2.6 Activation Function(Tanh and LeakyReLU)

In the generator, the Tanh activation function is employed for output normalization and ensuring pixel values are within the range of -1 to 1, making it suitable for image generation tasks. In the discriminator models, the Leaky Rectified Linear Unit (LeakyReLU) activation function is used. This activation function is applied to the outputs of convolutional layers within the downsample and upsample blocks. LeakyReLU is preferred in GANs because it helps mitigate the "dying ReLU" problem, allowing neurons to have non-zero gradients for both positive and negative inputs, which is important for training stability.

4.2.7 Generator Model plot

In this section the generator model is showed through a plot 4.3.

4.2.8 Discriminator model plot

In this section the discriminator model is showed through a plot 4.4.

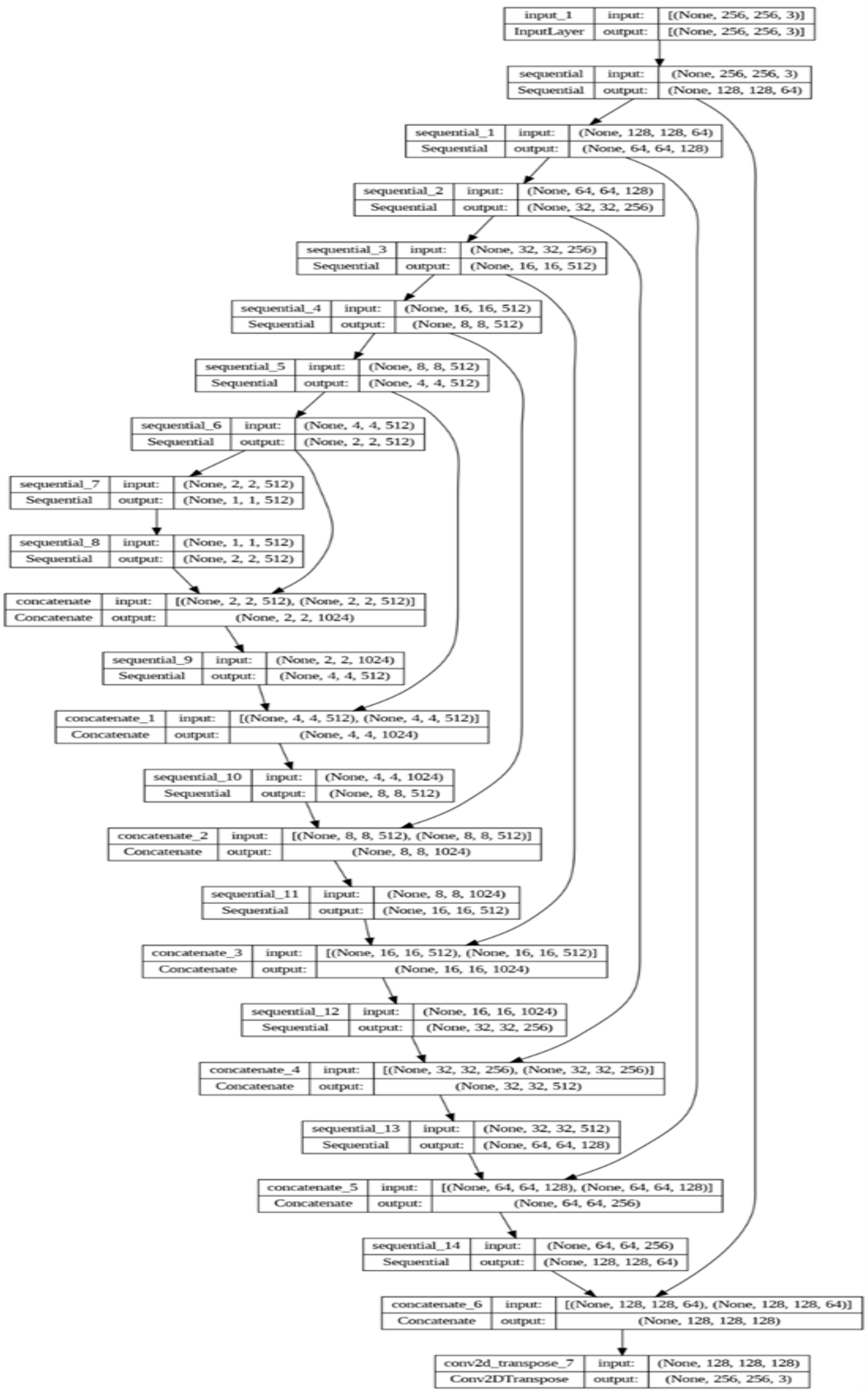

input_1 | InputLayer | input: [(None, 256, 256, 3)] | output: [(None, 256, 256, 3)]
sequential | Sequential | input: (None, 256, 256, 3) | output: (None, 128, 128, 64)
sequential_1 | Sequential | input: (None, 128, 128, 64) | output: (None, 64, 64, 128)
sequential_2 | Sequential | input: (None, 64, 64, 128) | output: (None, 32, 32, 256)
sequential_3 | Sequential | input: (None, 32, 32, 256) | output: (None, 16, 16, 512)
sequential_4 | Sequential | input: (None, 16, 16, 512) | output: (None, 8, 8, 512)
sequential_5 | Sequential | input: (None, 8, 8, 512) | output: (None, 4, 4, 512)
sequential_6 | Sequential | input: (None, 4, 4, 512) | output: (None, 2, 2, 512)
sequential_7 | Sequential | input: (None, 2, 2, 512) | output: (None, 1, 1, 512)
sequential_8 | Sequential | input: (None, 1, 1, 512) | output: (None, 2, 2, 512)
concatenate | Concatenate | input: [(None, 2, 2, 512), (None, 2, 2, 512)] | output: (None, 2, 2, 1024)
sequential_9 | Sequential | input: (None, 2, 2, 1024) | output: (None, 4, 4, 512)
concatenate_1 | Concatenate | input: [(None, 4, 4, 512), (None, 4, 4, 512)] | output: (None, 4, 4, 1024)
sequential_10 | Sequential | input: (None, 4, 4, 1024) | output: (None, 8, 8, 512)
concatenate_2 | Concatenate | input: [(None, 8, 8, 512), (None, 8, 8, 512)] | output: (None, 8, 8, 1024)
sequential_11 | Sequential | input: (None, 8, 8, 1024) | output: (None, 16, 16, 512)
concatenate_3 | Concatenate | input: [(None, 16, 16, 512), (None, 16, 16, 512)] | output: (None, 16, 16, 1024)
sequential_12 | Sequential | input: (None, 16, 16, 1024) | output: (None, 32, 32, 256)
concatenate_4 | Concatenate | input: [(None, 32, 32, 256), (None, 32, 32, 256)] | output: (None, 32, 32, 512)
sequential_13 | Sequential | input: (None, 32, 32, 512) | output: (None, 64, 64, 128)
concatenate_5 | Concatenate | input: [(None, 64, 64, 128), (None, 64, 64, 128)] | output: (None, 64, 64, 256)
sequential_14 | Sequential | input: (None, 64, 64, 256) | output: (None, 128, 128, 64)
concatenate_6 | Concatenate | input: [(None, 128, 128, 64), (None, 128, 128, 64)] | output: (None, 128, 128, 128)
conv2d_transpose_7 | Conv2DTranspose | input: (None, 128, 128, 128) | output: (None, 256, 256, 3)

Figure 4.3: Generator Model

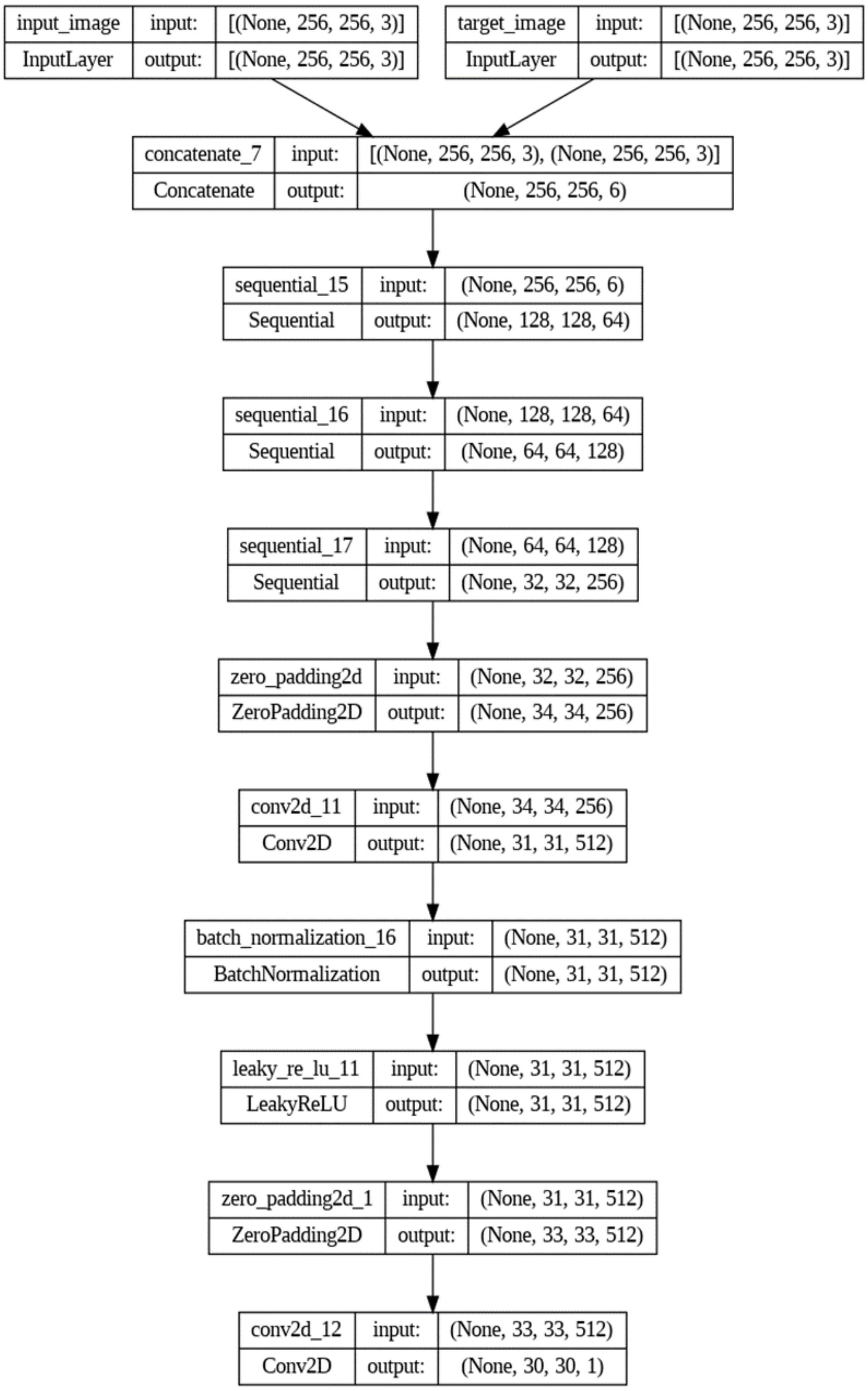

input_image | InputLayer | input: [(None, 256, 256, 3)] | output: [(None, 256, 256, 3)]
target_image | InputLayer | input: [(None, 256, 256, 3)] | output: [(None, 256, 256, 3)]
concatenate_7 | Concatenate | input: [(None, 256, 256, 3), (None, 256, 256, 3)] | output: (None, 256, 256, 6)
sequential_15 | Sequential | input: (None, 256, 256, 6) | output: (None, 128, 128, 64)
sequential_16 | Sequential | input: (None, 128, 128, 64) | output: (None, 64, 64, 128)
sequential_17 | Sequential | input: (None, 64, 64, 128) | output: (None, 32, 32, 256)
zero_padding2d | ZeroPadding2D | input: (None, 32, 32, 256) | output: (None, 34, 34, 256)
conv2d_11 | Conv2D | input: (None, 34, 34, 256) | output: (None, 31, 31, 512)
batch_normalization_16 | BatchNormalization | input: (None, 31, 31, 512) | output: (None, 31, 31, 512)
leaky_re_lu_11 | LeakyReLU | input: (None, 31, 31, 512) | output: (None, 31, 31, 512)
zero_padding2d_1 | ZeroPadding2D | input: (None, 31, 31, 512) | output: (None, 33, 33, 512)
conv2d_12 | Conv2D | input: (None, 33, 33, 512) | output: (None, 30, 30, 1)

Figure 4.4: Discriminator model

Chapter 5

Evaluation

5.1 Evaluation

In this section, we assess the performance of the Conditional Generative Adversarial Network (cGAN) model for Image Restoration. The primary objective is to evaluate the quality of the generated color images and the model's ability to faithfully reproduce colors in grayscale inputs.

5.2 Dataset

We conducted our experiments using a dataset [ ] comprising a diverse range of grayscale images, each paired with its corresponding ground truth color image. The dataset was split into training and testing subsets. We minizine the dataset to reduce the complexity and resource usage .

5.3 Quantitative Metrics

5.3.1 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio is a widely used metric for evaluating image quality. It measures the similarity between the generated color images and the ground truth images. Higher PSNR values indicate better image quality.

5.3.2 Structural Similarity Index (SSIM)

The Structural Similarity Index measures the structural similarity between two images. It assesses the perceptual quality of the generated color images in comparison to the ground truth images. Higher SSIM values correspond to higher image fidelity.

5.3.3 Qualitative Evaluation

In addition to quantitative metrics, we performed a qualitative assessment by visually inspecting the generated color images. We present sample results and conduct a visual comparison to gauge the model's ability to preserve details and colors from the input grayscale images.

5.3.4 Results

The GAN model was trained over a specific number of epochs, and the performance metrics were computed on the testing dataset to assess the model's image translation capabilities.

5.3.5 Peak Signal-to-Noise Ratio (PSNR)

In PSNR value normally ranges between 46.225-46.2400 for out 5 test samples. Corresponding figure is 5.1

5.3.6 Structural Similarity Index (SSIM)

Corresponding figure of Structural Similarity Index (SSIM) is 5.2

5.4 Discussion

The quality measurement in has been done with 5 train and test data from out dataset .The evaluation results suggest that the cGAN model has demonstrated significant proficiency in Image Restoration. The high PSNR and SSIM values signify the quality and

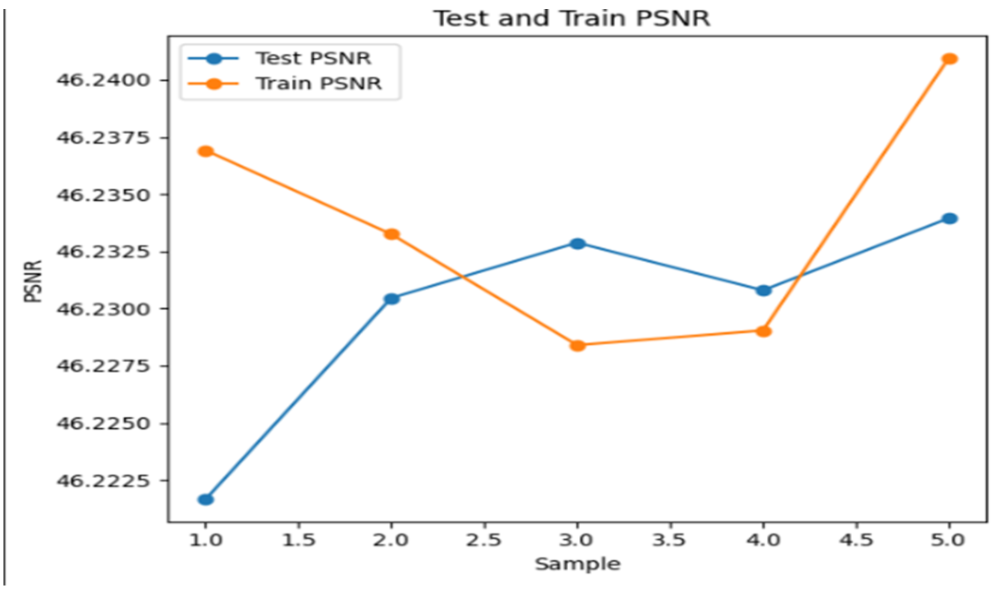


Figure 5.1: Peak Signal-to-Noise Ratio

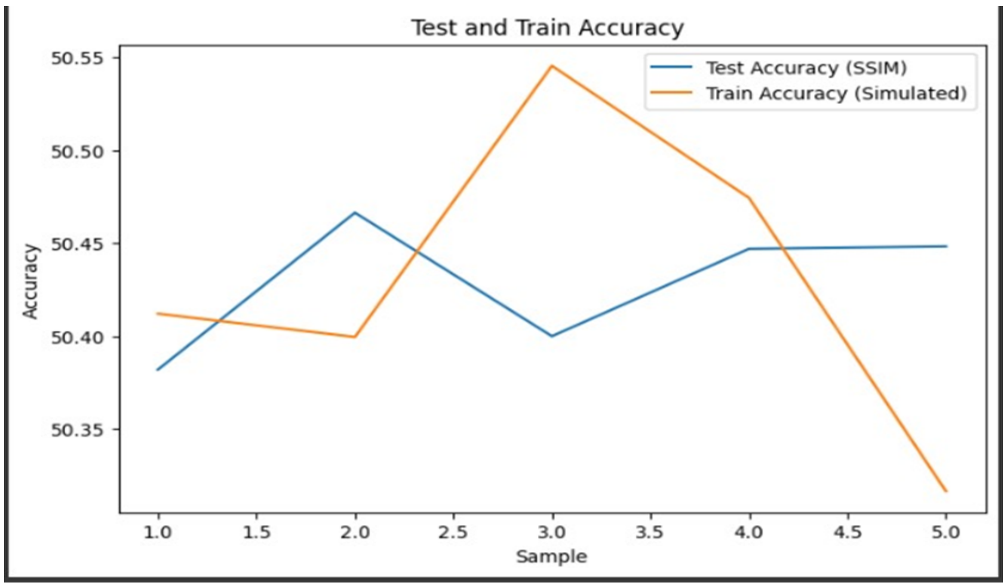


Figure 5.2: Structural Similarity Index

similarity of the generated color images to the ground truth images. The visual assessment further corroborates the model's capability to produce realistic colorizations.

5.5 Outcome

In this section we will discuss how the project actually works in real life scenarios:

5.5.1 Step 1: Model Training and Saving

After successfully training our deep learning model, we carefully saved the model to preserve its learned capabilities. This model holds the power to restore and enhance images, transforming pixelated or degraded visuals into their pristine forms.

5.5.2 Step 2: Development of a Web Interface

In the pursuit of making this image restoration service accessible to a wider audience, we embarked on the development of a user-friendly web interface. Leveraging the Flask framework, we crafted a seamless user experience for image correction and enhancement.

5.5.3 Step 3: Web Application Launch

Upon execution of the app.py script, the web application springs to life, ready to serve users at http://127.0.0.1:5000. This development server hosts the main web page where users can interact with the image restoration system.

5.5.4 Step 4: Image Selection

With the web application at your disposal, the next step involves selecting a degraded or pixelated image residing on your desktop. This image, representing your canvas for restoration, carries the potential to be transformed into a pristine masterpiece.

5.5.5 Step 5: User Interaction

The elegantly designed webpage provides a seamless user experience. After meticulously selecting the image to be restored (e.g., 133.jpg), users are invited to initiate the image correction process by clicking the 'Upload and Restore' button.

5.5.6 Step 6: Image Restoration

With the user's command, the selected image is uploaded to the web application, signaling the beginning of the restoration journey. Behind the scenes, our trained model [generator

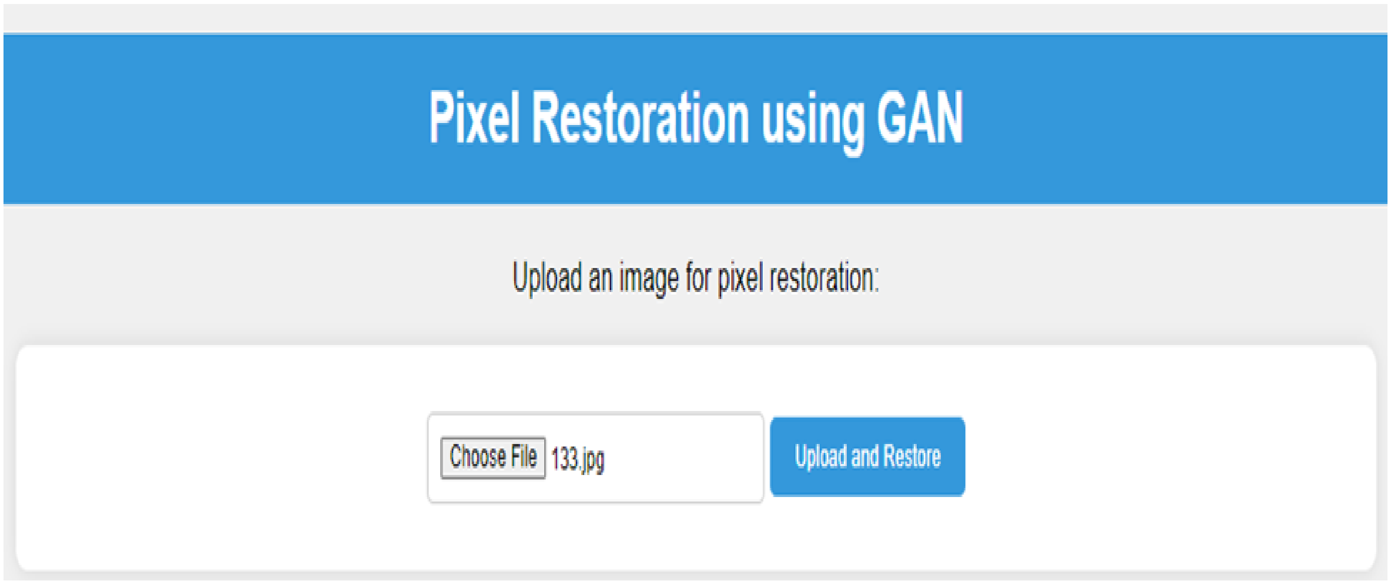

Pixel Restoration using GAN
Upload an image for pixel restoration:
Choose File
133.jpg
Upload and Restore

Figure 5.3: Image Selection

model] takes charge, applying its deep learning prowess to enhance and restore the image.

5.5.7 Step 7: Dual Image Display

Upon successful completion of the image correction process, users are presented with a visual representation of the transformation. The web interface showcases both the original input image and the restored output image side by side, allowing users to appreciate the remarkable difference.

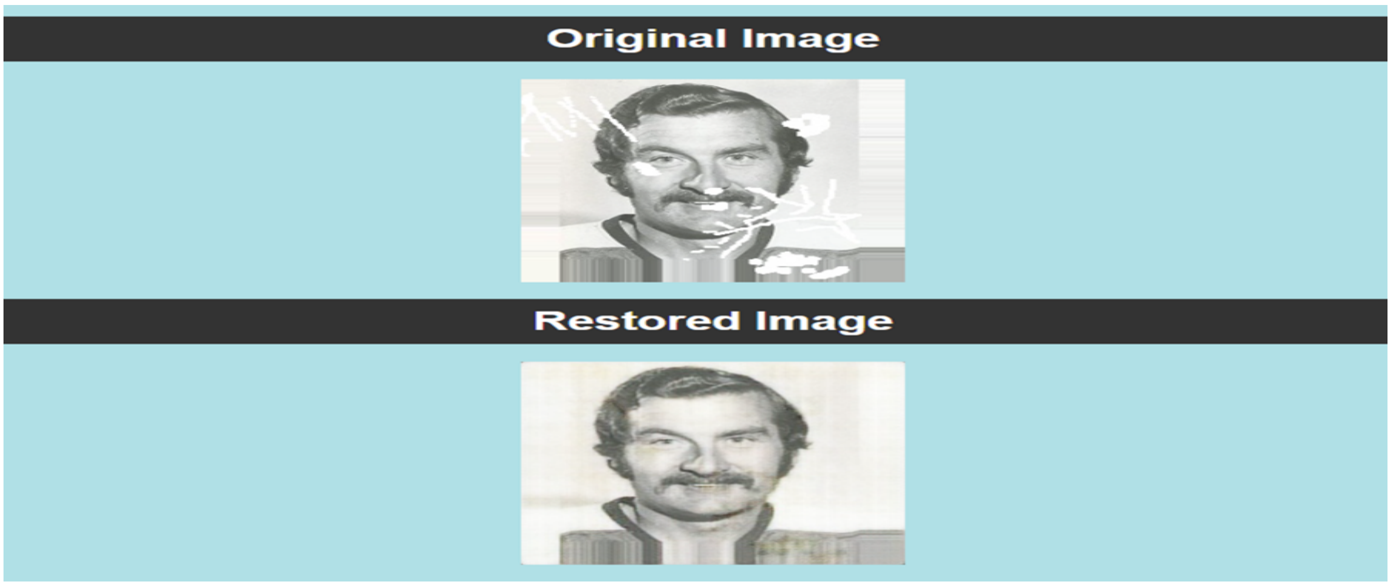


Figure 5.4: Dual Image Display

5.5.8 Step 8: Image Storage

As a testament to our commitment to user convenience, the web application automatically archives the last input image as ”uploadedimage” and the corresponding output image as ”restoredimage” in the ’static’ folder. This ensures that users can easily access and retrieve their corrected images for future reference and use. In summary, our web interface,

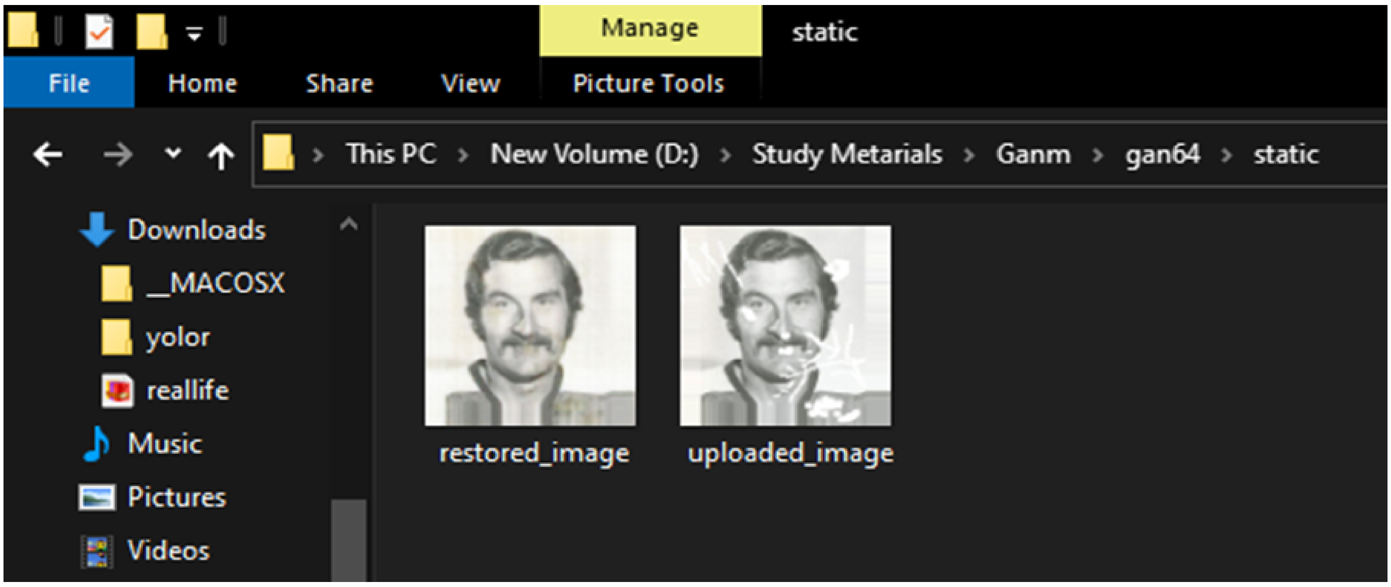


Figure 5.5: Image Storage

seamlessly integrated with our model, empowers users to effortlessly correct and enhance images with a few simple clicks, providing a transformative experience in the world of digital imagery.

Chapter 6

Summary and Reflections

6.1 Limitations

[1]Occasional Color Artifacts: The current implementation may exhibit sporadic color artifacts in generated images, requiring further refinement.
[2]Slight Color Deviations: Some color deviations from ground truth images may be observed, which could impact overall image fidelity.
[3]Challenges with Specific Image Content: The model may face challenges when dealing with complex image content, such as intricate textures or irregular patterns.

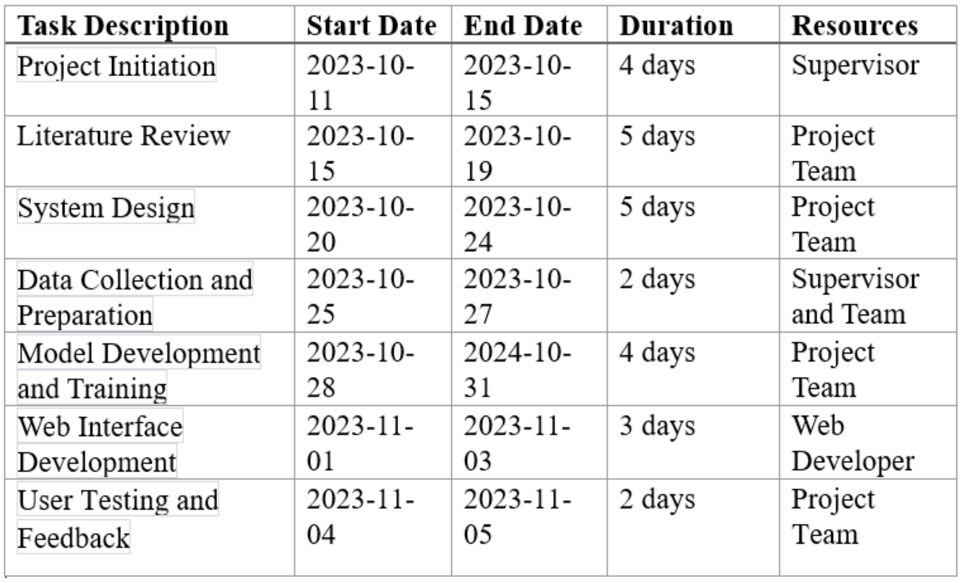
6.2 Future Work

[1]Model Architecture Refinement: Investigate and fine-tune the model architecture, including the depth and width of neural networks, to further improve colorization accuracy.
[2]Exploration of Alternative Loss Functions: Experiment with alternative loss functions, such as perceptual loss or adversarial loss variations, to optimize image translation performance.
[3]Conditional Variations: Explore different conditional setups, such as utilizing semantic masks or multi-modal inputs, to enhance the model's adaptability to diverse image types.
[4]Dataset Augmentation and Domain Adaptation: Augment the dataset with a broader

range of image variations and investigate domain adaptation techniques to improve model performance on specific image types or domains.

6.3 Project management

The work plan of the project is shown in the figure 6.1



Task Description	Start Date	End Date	Duration	Resources
Project Initiation	2023-10-11	2023-10-15	4 days	Supervisor
Literature Review	2023-10-15	2023-10-19	5 days	Project Team
System Design	2023-10-20	2023-10-24	5 days	Project Team
Data Collection and Preparation	2023-10-25	2023-10-27	2 days	Supervisor and Team
Model Development and Training	2023-10-28	2024-10-31	4 days	Project Team
Web Interface Development	2023-11-01	2023-11-03	3 days	Web Developer
User Testing and Feedback	2023-11-04	2023-11-05	2 days	Project Team

Figure 6.1: Project management

6.4 Contributions and reflections

Together, as a collaborative team, the three of us made significant contributions to the project in terms of code development, project design, and presentation.

Bibliography

[1] Jin, X., Hu, Y., and Zhang, C.-Y. Image restoration method based on gan and multi-scale feature fusion. In *2020 Chinese Control And Decision Conference (CCDC)* (2020), IEEE, pp. 2305–2310.

[2] Jung, W., Jung, D., Kim, B., Lee, S., Rhee, W., and Ahn, J. H. Restructuring batch normalization to accelerate cnn training. In *Proceedings of Machine Learning and Systems* (2019), A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, pp. 14–26.

[3] Kishore, A., Kumar, A., and Dang, N. Enhanced image restoration by gans using game theory. *Procedia Computer Science 173* (2020), 225–233.

[4] Li, S., Fan, R., Lei, G., Yue, G., and Hou, C. A two-channel convolutional neural network for image super-resolution. *Neurocomputing 275* (2018), 267–277.

[5] Yu, H., Liu, Y., He, S., Jiang, P., Xin, J., and Wen, J. A practical generative adversarial network architecture for restoring damaged character photographs. *Neurocomputing 423* (2021), 590–600.