TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

A MINOR PROJECT REPORT
ON
”IMAGE COLORIZATION USING DEEP LEARNING AND cGANs”

BY

MD ASTAFAR ALAM(PUR078BCT049)

RESHMI JHA(PUR078BCT066)

RITESH SAHANI(PUR078BCT068)

SUBASH KUMAR YADAV (PUR078BCTO87)

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PURWANCHAL CAMPUS

DHARAN, NEPAL

MARCH,2025

ACKNOWLEDGEMENT

We express our sincere gratitude to Mr. Pukar Karki, Deputy Head of the Department of Electronics and Computer Engineering, for providing us with the opportunity and resources to undertake this minor project. We are deeply thankful to our project supervisor, Assistant Prof. Binaylal Shrestha, for his invaluable guidance, constant encouragement, and insightful feedback throughout the development of this project on **"IMAGE COLORIZATION"** using conditional Generative Adversarial Networks (cGANs).

We also extend our appreciation to all the faculty members of the Electronics and Computer Department for their support and technical expertise, which greatly enriched our learning experience. Special thanks to our classmates and friends for their constructive suggestions and motivation. Finally, we acknowledge the unwavering support of our families, whose encouragement kept us motivated.

This project would not have been possible without the collective contributions of everyone involved.

ABSTRACT

This project presents an automated image colorization system using Conditional Generative Adversarial Networks (cGANs), addressing the challenge of transforming grayscale images into realistic color outputs. Leveraging a U-Net architecture with a ResNet18 backbone as the Generator and a 70x70 Patch Discriminator, the model learns to predict chromatic values (ab channels) from luminance (L channel) inputs in the L*a*b* color space. Trained on a dataset of paired grayscale-color images, the system employs a hybrid loss combining Binary Cross-Entropy with Logits Loss (adversarial) and L1 Loss (pixel-wise accuracy), with the Generator pretrained for 20 epochs to enhance convergence. Results demonstrate effective colorization, with training PSNR reaching 34 dB and test PSNR stabilizing at 23 dB, indicating overfitting challenges. Qualitative outputs show vibrant, context-aware colorization, though minor over-saturation occurs. This work contributes to automated image restoration and digital media, with future potential for improved generalization and high-resolution scaling.

TABLE OF CONTENTS

ACKNOWLEDGEMENT **i**

ABSTRACT **ii**

LIST OF FIGURES **vi**

LIST OF ABBREVIATIONS **vii**

1 INTRODUCTION **1**

1.1 Background 1

1.2 Gap Identification 1

1.3 Motivation 1

1.4 Objectives 1

2 RELATED THEORY **3**

2.1 Convolutional Neural Networks (CNNs) 3

2.2 Generative Adversarial Networks (GANs) 3

2.3 Conditional GANs (cGANs) 4

2.4 U-Net 5

2.5 ResNet (Residual Network) 7

2.6 Patch Discriminator 7

2.7 Loss Functions 9

3 LITERATURE REVIEW **10**

3.1 Early Colorization Techniques 10

3.2 Learning-Based Approaches 10

3.3 GANs in Image Colorization 11

3.4 Conditional GANs (cGANs) 11
3.5 Advancements and Variants 11
3.6 Gaps and Relevance to Our Work 12
3.7 Conclusion 12

4 METHADOLOGY **13**

4.1 Dataset Collection 13
4.2 Data processing 13
4.3 Dataset Preparation 13
4.4 Model Architecture 14
4.5 Training Process 14
4.6 Evaluation Metrics 16
4.7 Implementation Details 18
4.8 Web Application Integration 18
4.8.1 System Overview 18
4.8.2 Use Case Diagram 18
4.8.3 Request Response Diagram 18

5 SYSTEM DESIGN **19**

5.1 System Overview 19
5.2 Use Case Diagram 20
5.3 Request Response Diagram 21

6 RESULT AND CONCLUSION **22**

6.1 Results 22
6.2 Conclusion 22

REFERENCES **23**

LIST OF FIGURES

Figure 2.1: Overall architecture of the cGAN for image colorization, featuring a U-Net Generator and Patch Discriminator. 5

Figure 2.2: U-Net architecture with skip connection and 8 layer deep. 6

Figure 2.3: Patch Discriminator architecture with 70×70 receptive fields, outputting a 30×30 realism map. 8

Figure 4.1: U-Net architecture with ResNet18 as backbone(encoder). 14

Figure 4.2: Discriminator training process. 15

Figure 4.3: Generator training process. 16

Figure 4.4: PSNR behaviour during training and validation. 17

Figure 4.5: Total GAN loss behaviour during training and validation. 17

Figure 5.1: system overview. 19

Figure 5.2: Use Case Diagram 20

Figure 5.3: Request Response Diagram 21

Figure 6.1: Sample grayscale inputs and their colorized outputs using the cGAN model, demonstrating realistic colorization. 22

LIST OF ABBREVIATIONS

API : Application Programming Interface

AI : Artificial Intelligence

ML : Machine Learning

CNN : Convolutional Neural Network

GAN : Generative Adversial Network

MSE : Mean Squared Error

GPU : Graphics Processing Unit

PSNR : Peak Signal-to-noise Ration

SSIM : Structural Similarity Index Measure

cGAN : conditional Generative Adversarial Networks

MAE : Mean Absolute Error

CHAPTER 1

INTRODUCTION

1.1 Background

Image colorization transforms grayscale images into vibrant colored versions, with applications in historical restoration, media enhancement, and digital art. Traditional methods involve manual effort or simplistic algorithms, often producing unrealistic results. Advances in deep learning, particularly conditional Generative Adversarial Networks (cGANs), offer a promising solution by automating this process with context-aware color generation.

1.2 Gap Identification

While existing deep learning approaches improve colorization, many struggle with realistic color coherence and generalization to diverse datasets. GAN-based methods show potential but often require careful tuning to avoid overfitting or unnatural outputs, a challenge we addressed in this project.

1.3 Motivation

The ability to revive gray-scale images with realistic colors has both practical and artistic value. Our motivation stems from harnessing cGANs generative power to create a robust, automated colorization tool, contributing to advancements in image processing while enhancing our technical skills.

1.4 Objectives

- Develop a cGAN model to colorize grayscale images automatically.
- To enhance usability, this project also developed a web application to interface with the cGAN colorization model, enabling users to upload grayscale images and view colorized outputs.The system overview of web application shown Figure 5.1

- Evaluate performance using PSNR (peak signal-to-noise ratio) as metric and analyze generalization on unseen data.

CHAPTER 2

RELATED THEORY

This chapter outlines the theoretical foundations underpinning our image colorization project using conditional Generative Adversarial Networks (cGANs). It covers the core concepts and components integral to our implementation, providing a basis for understanding the methodology and system design.

2.1 Convolutional Neural Networks (CNNs)

CNNs are a specialized subset of neural networks optimized for image-based tasks. Through convolutional layers, pooling layers, and activation functions, CNNs extract spatial features such as edges, patterns, and textures. Their hierarchical structure allows early layers to capture low-level features (e.g., edges) and deeper layers to identify high-level features (e.g., objects or scenes). This makes CNNs highly effective for image colorization, where understanding spatial and semantic context is critical for realistic color assignment.

2.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs), introduced by Goodfellow et al. in 2014, are a class of deep learning models designed for generative tasks. GANs consist of two neural networks: a Generator that synthesizes data from random noise or conditioned inputs, and a Discriminator that distinguishes between real and generated data. These networks are trained adversarially as in a minimax game, where the Generator improves by attempting to "fool" the Discriminator, and the Discriminator enhances its ability to identify fakes. Mathematically, the objective is defined as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

where x represents real data, z denotes noise, and D and G are the Discriminator and Generator functions, respectively. As we can see, in this equation generation tries to reduce the total loss meanwhile discriminator approach to increase it .

2.3 Conditional GANs (cGANs)

Conditional GANs (cGANs) extend the GAN framework by conditioning both the Generator and Discriminator on additional input, such as a grayscale image in our project. This transforms the problem into an image-to-image translation task, where the Generator learns a mapping $G : X \to Y$ (grayscale X to color Y) guided by the Discriminator's feedback. Here , for our model we remove the noise and feed grayscale as only input to generator, based on the input it will map color to each pixel. The loss function for cGANs incorporates this conditioning:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x,y\sim p_{\text{data}}(x,y)}[\log D(x,y)] + \mathbb{E}_{x\sim p_x(x),z\sim p_z(z)}[\log(1-D(x,G(x,z)))]$$

Here, x is the grayscale input, and y is the corresponding real color image, enabling context-aware colorization.

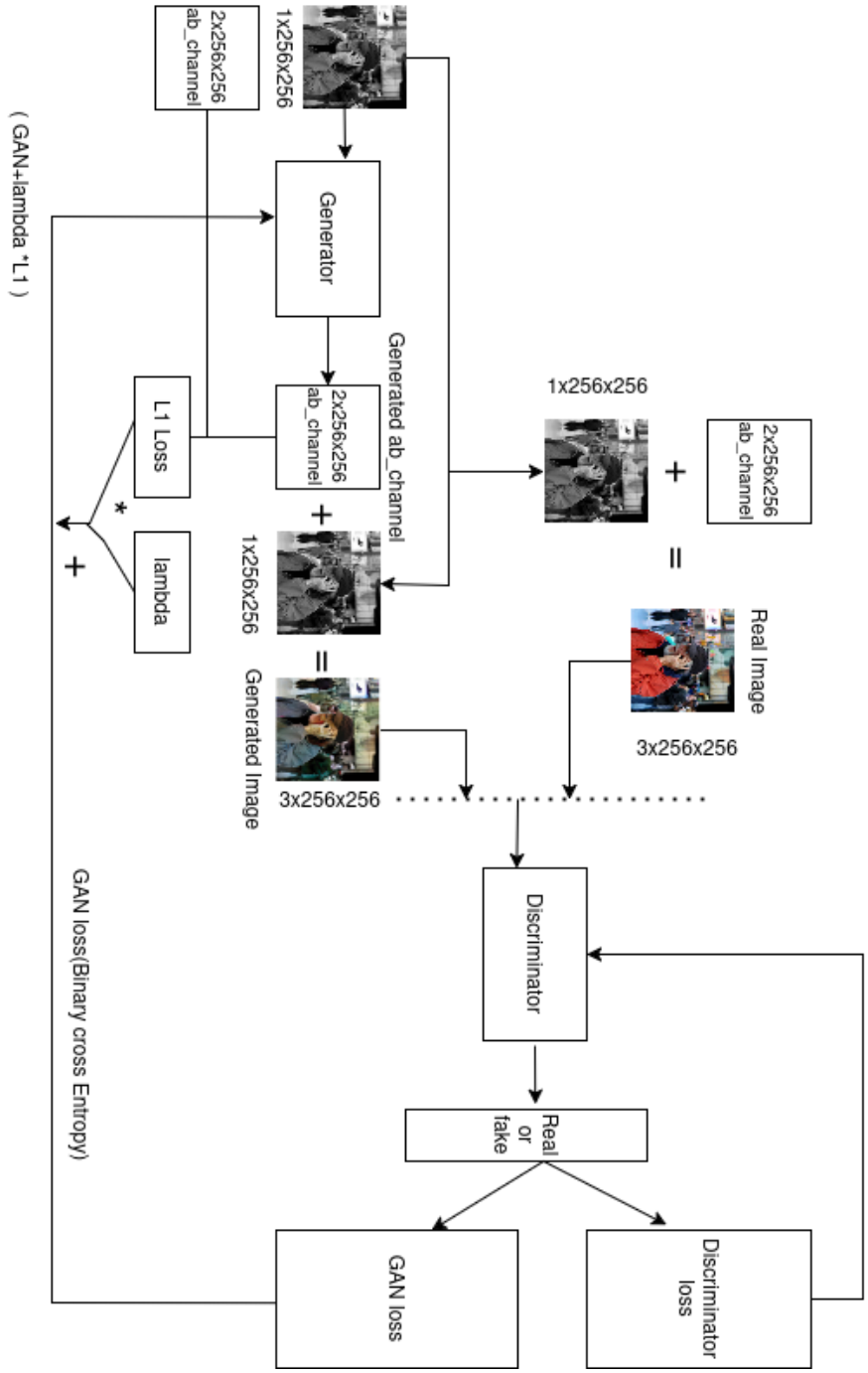

Figure 2.1: Overall architecture of the cGAN for image colorization, featuring a U-Net Generator and Patch Discriminator.

2.4 U-Net

U-Net is a convolutional neural network architecture featuring an encoder-decoder structure with skip connections. The encoder compresses the grayscale input into a latent representation, while the decoder reconstructs it into a colorized output. Skip connec-

tions between corresponding encoder and decoder layers preserve spatial details lost during downsampling, making U-Net particularly suited for pixel-level tasks like colorizatio, as used in our Generator design.

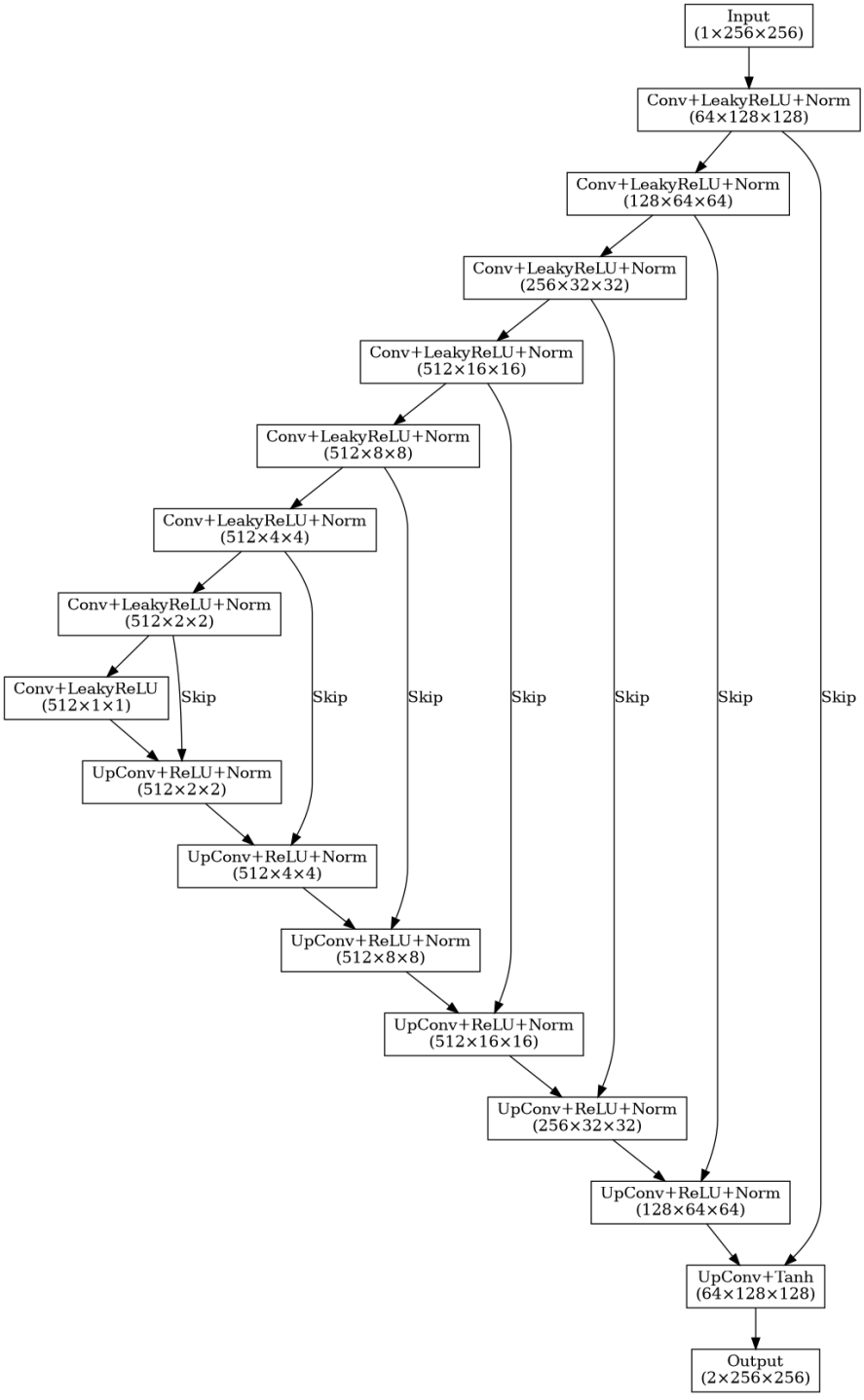

Figure 2.2: U-Net architecture with skip connection and 8 layer deep.

2.5 ResNet (Residual Network)

We employ ResNet18 as the backbone of the U-Net's encoder. ResNet, or Residual Network, introduces residual blocks with shortcut connections, defined as $y = F(x) + x$, where $F(x)$ is the learned residual function. This design mitigates vanishing gradient problems during training and enhances feature extraction, which is essential for capturing the intricate(complexly arranged) context of images in our colorization task.

2.6 Patch Discriminator

In simple, discriminator is a binary classification model which is used to classify either the image provided to it is real image or the one which is made up using generator. In this way it can be help for the generator to generate more realistic images,which is the goal of our project. Unlike traditional Discriminators that output a single real/fake score for an entire image, the Patch Discriminator assesses realism over local 70x70 patches. Inspired by Isola et al. (2016), this approach emphasizes high-frequency details and textures, encouraging the Generator to produce locally coherent colors rather than globally averaged ones. This is a key component of our cGANs implementation, enhancing output quality.

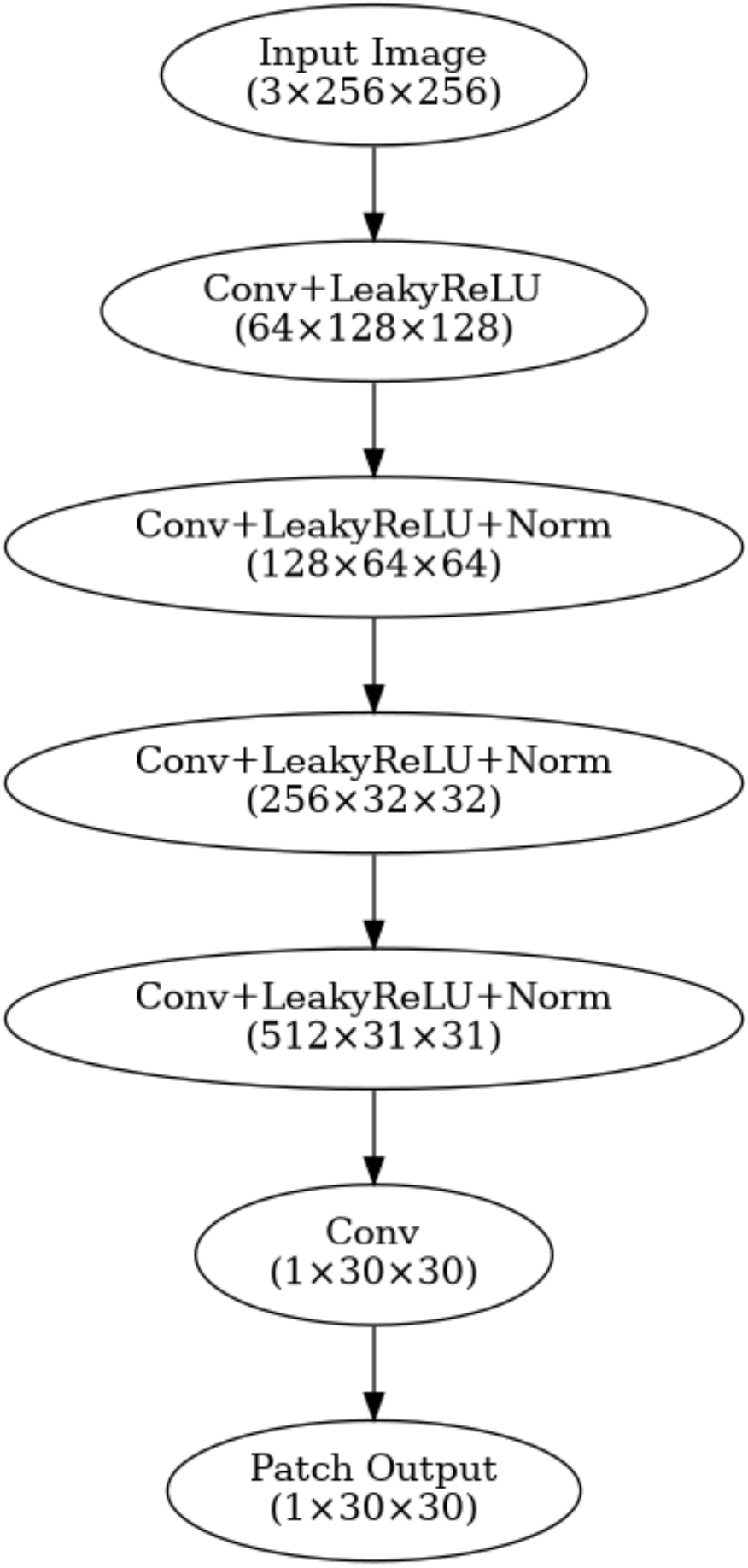

Figure 2.3: Patch Discriminator architecture with 70×70 receptive fields, outputting a 30×30 realism map.

2.7 Loss Functions

The Generator minimizes a hybrid loss:

BCEWithLogitsLoss: Measures the adversarial loss, encouraging the Generator to produce outputs indistinguishable from real images.

L1 Loss: Ensures pixel-wise accuracy between the generated and ground-truth color images, defined as $L1 = \mathbb{E}||y - G(x)||$.The combined loss balances realism

The interplay of these components enables our cGANs to learn complex color distributions from grayscale inputs, leveraging pretraining (20 epochs for the Generator) to accelerate convergence and adversarial training to refine outputs. This theoretical foundation underpins our methodology and system design, bridging traditional image processing with modern generative AI techniques.

CHAPTER 3

LITERATURE REVIEW

The field of image colorization has evolved significantly over the years, transitioning from manual techniques to sophisticated deep learning approaches. This chapter reviews the progression of colorization methods, culminating(highly used) in the adoption of Generative Adversarial Networks (GANs) and their conditional variants (cGANs), which form the foundation of our project.

3.1 Early Colorization Techniques

Traditional image colorization relied heavily on human intervention. Artists manually assigned colors to grayscale images using tools like Adobe Photoshop, a process that was both time-consuming and skill-intensive. Early automated methods, introduced color transfer techniques where colors were borrowed from a reference image based on luminance matching. However, these approaches struggled with semantic accuracy, often producing inconsistent or unrealistic results due to their reliance on simplistic statistical mappings rather than contextual understanding.

3.2 Learning-Based Approaches

The rise of machine learning brought a shift toward data-driven colorization. Early methods introduced semi-automatic techniques where users provided color hints, and an optimization algorithm propagated these hints across the image. While this approach reduced manual effort, it still required user input, limiting full automation. With advancements in deep learning, fully automated methods emerged, using neural networks to predict colors from grayscale images. Some approaches incorporated hand-crafted features such as edges and textures to guide the colorization process. However, these early neural network-based methods often produced muted or uniform colors, lacking the vibrancy and realism needed for practical applications.

3.3 GANs in Image Colorization

Generative Adversarial Networks (GANs) introduced a groundbreaking approach to generative modeling by employing an adversarial framework, where a Generator competes against a Discriminator. This method has been particularly effective in image-related tasks. Early applications of GANs to colorization focused on training models to generate realistic color distributions from grayscale images. The adversarial loss played a crucial role in enhancing realism by encouraging outputs that closely resembled real-world color patterns. However, standard GANs lacked control over the generated outputs, often resulting in arbitrary color assignments.

3.4 Conditional GANs (cGANs)

Conditional GANs (cGANs), proposed by Mirza and Osindero (2014), addressed this limitation by conditioning the Generator and Discriminator on additional input, such as grayscale images. Isola et al. (2016) extended this concept with their Pix2Pix framework, a cGAN-based model for general image-to-image translation. They paired a U-Net Generator with a Patch Discriminator, using a hybrid loss combining adversarial (BCE) and pixel-wise (L1) terms. Applied to colorization, Pix2Pix achieved state-of-the-art results, producing vibrant, context-aware colors. Our project draws inspiration from this work, adopting a similar U-Net architecture and Patch Discriminator for local realism.

3.5 Advancements and Variants

Subsequent research refined cGAN-based colorization. Cao et al. (2017) introduced a fully convolutional network with multi-scale feature extraction, improving color consistency across large images. Nazeri et al. (2018) proposed ChromaGAN, which added a perceptual loss to enhance visual quality, addressing over-saturation issues common in earlier models. Meanwhile, residual networks (ResNet), introduced by He et al. (2016), became popular backbones for Generators, mitigating training instability and boosting feature extraction. Our use of ResNet18 as the U-Net backbone aligns with this trend, leveraging residual connections to capture intricate image details.

3.6 Gaps and Relevance to Our Work

Despite these advancements, challenges remain. Many models overfit to training datasets, as evidenced by saturated test metrics (e.g., PSNR), a phenomenon we observed in our experiments. Balancing adversarial and pixel-wise losses is also critical, as over-reliance on L1 can dull creativity, while weak adversarial supervision may yield unrealistic hues. Our project addresses these gaps by pretraining the Generator (20 epochs) to stabilize early training and employing a 70x70 Patch Discriminator to focus on local coherence, building on the strengths of Pix2Pix while tailoring the approach to our specific colorization goals.

3.7 Conclusion

The literature highlights a clear progression from manual to automated colorization, with cGANs emerging as a powerful tool for realistic image synthesis. By integrating established techniques—U-Net for spatial detail, ResNet for feature robustness, and Patch Discriminators for texture fidelity—our work contributes to this evolving field, aiming to balance generalization and realism in grayscale-to-color translation.

CHAPTER 4

METHADOLOGY

This chapter details the approach taken to implement our image colorization system using conditional Generative Adversarial Networks (cGANs). It outlines the dataset preparation, model architecture, and training process employed to achieve realistic colorization of grayscale images.

4.1 Dataset Collection

We downloaded the small version of COCO dataset,the dataset we used for training and validation is 2017 COCO validation dataset,we used fastai API to download approx 20k of image samples of which 10k used for our model.

4.2 Data processing

Once we have dataset collected, firstly we convert the images from RGB color space to CIELAB color space. In CIELAB color space, L*(lightness) Represents the brightness of a color, ranging from 0 (black) to 100 (white).Where as A* represents the green-red axis, with negative values indicating green and positive values indicating red and B* represents the blue-yellow axis, with negative values indicating blue and positive values indicating yellow. We do the image preprocessing like converting from RGB to LAB and normalization overcome the complexity of computation by our model. Predicting two channel for each pixel would be lot easier thant predicting three channel for each pixel.

In our project model predict the missing ab* channel for each pixel in the given L* channel input image, after the prediction we add the predicted output with input and convert image into RGB as our original format.

4.3 Dataset Preparation

After the preprocessing of collected data, we split dataset into training set and testing set. Moving forward we used custom DataLoader to load trainig and validation dataset

with batch size of 16.

4.4 Model Architecture

Our cGAN comprises two core components tailored for image-to-image translation. The Generator is a U-Net with a ResNet18 backbone, leveraging skip connections to retain spatial details and residual blocks to enhance feature extraction. The Discriminator is a 70x70 Patch Discriminator, assessing realism over local regions rather than the entire image, promoting texture coherence. This architecture draws from established designs(i.e. fastai API) while aligning with our goal of realistic colorization.
generator architecture:

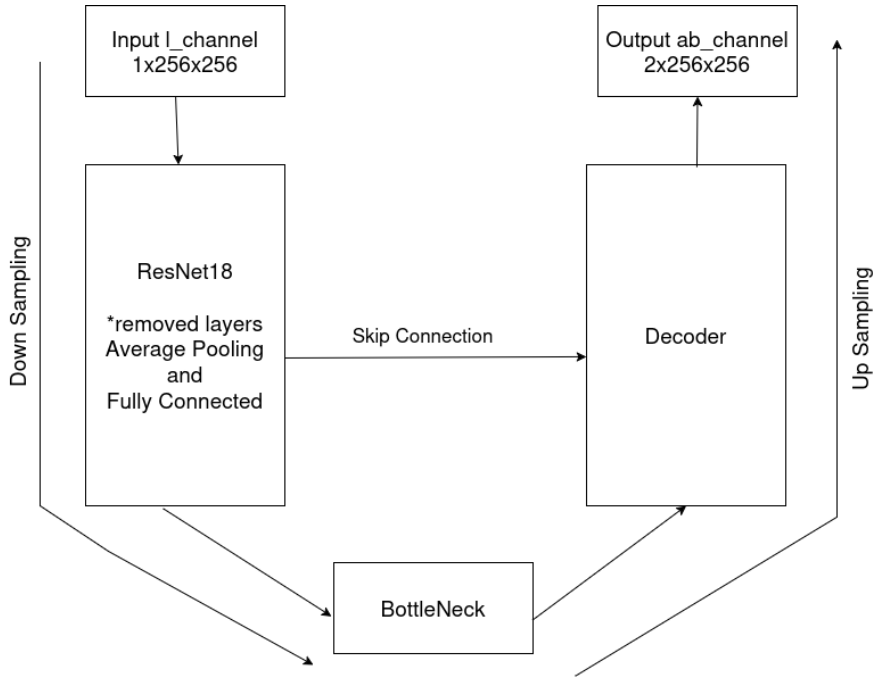

Figure 4.1: U-Net architecture with ResNet18 as backbone(encoder).

4.5 Training Process

The training was conducted in two phases. First, the Generator was pretrained for 20 epochs using only the L1 loss to establish a baseline mapping from grayscale to color images. Subsequently, adversarial training commenced, combining BCEWithLogitsLoss (adversarial) and L1 loss (pixel-wise) to optimize the Generator, while the Discriminator used BCEWithLogitsLoss alone. We employed the Adam optimizer with

a learning rate of 0.0002 and $\beta_1 = 0.5$, training for 20 epochs until convergence or PSNR saturation was observed.

Generator Pretraining:

We followed the same process in loop for 20 epochs on 10,000 COCO-SAMPLE datasets, input grayscale image –cGAN generator –ab-channel–L1 loss –gradient descent generator

Here, we have broken down the adversarial training process into two parts for a better understanding of the training process, even though GAN training occurs adversarially in our model.

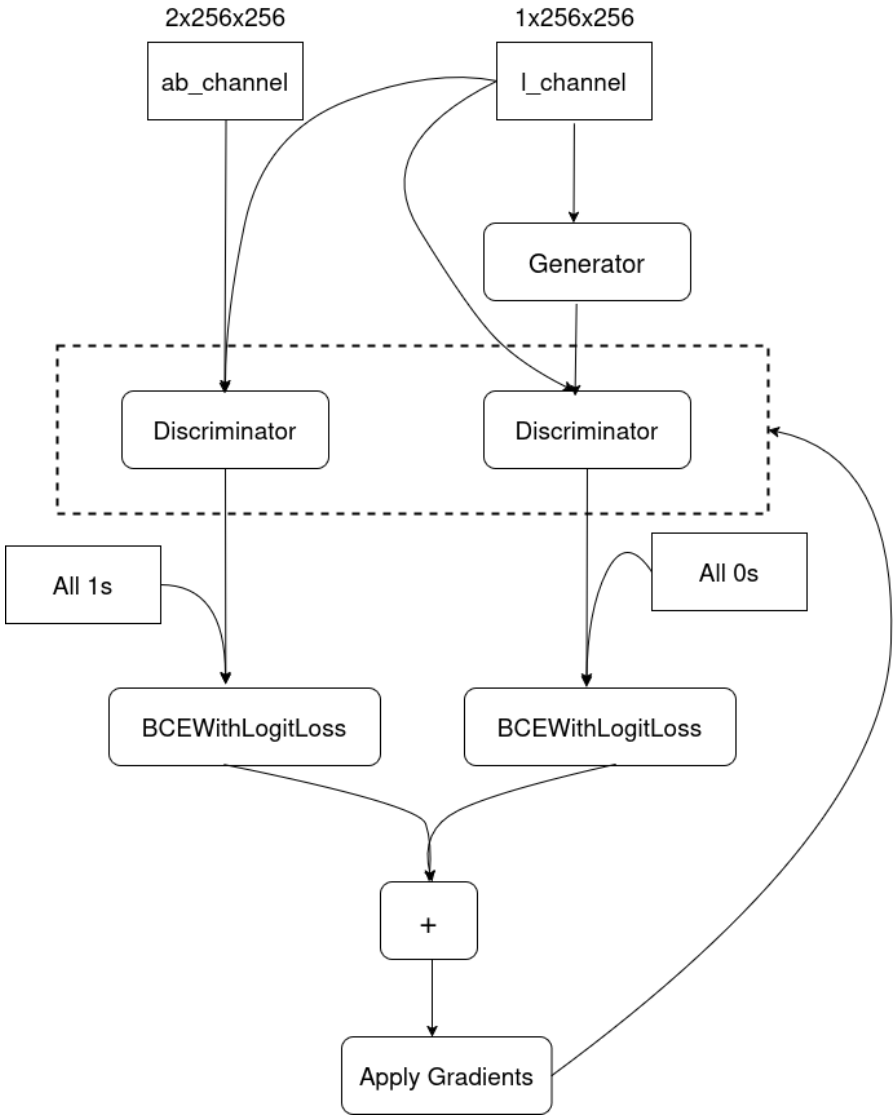

Figure 4.2: Discriminator training process.

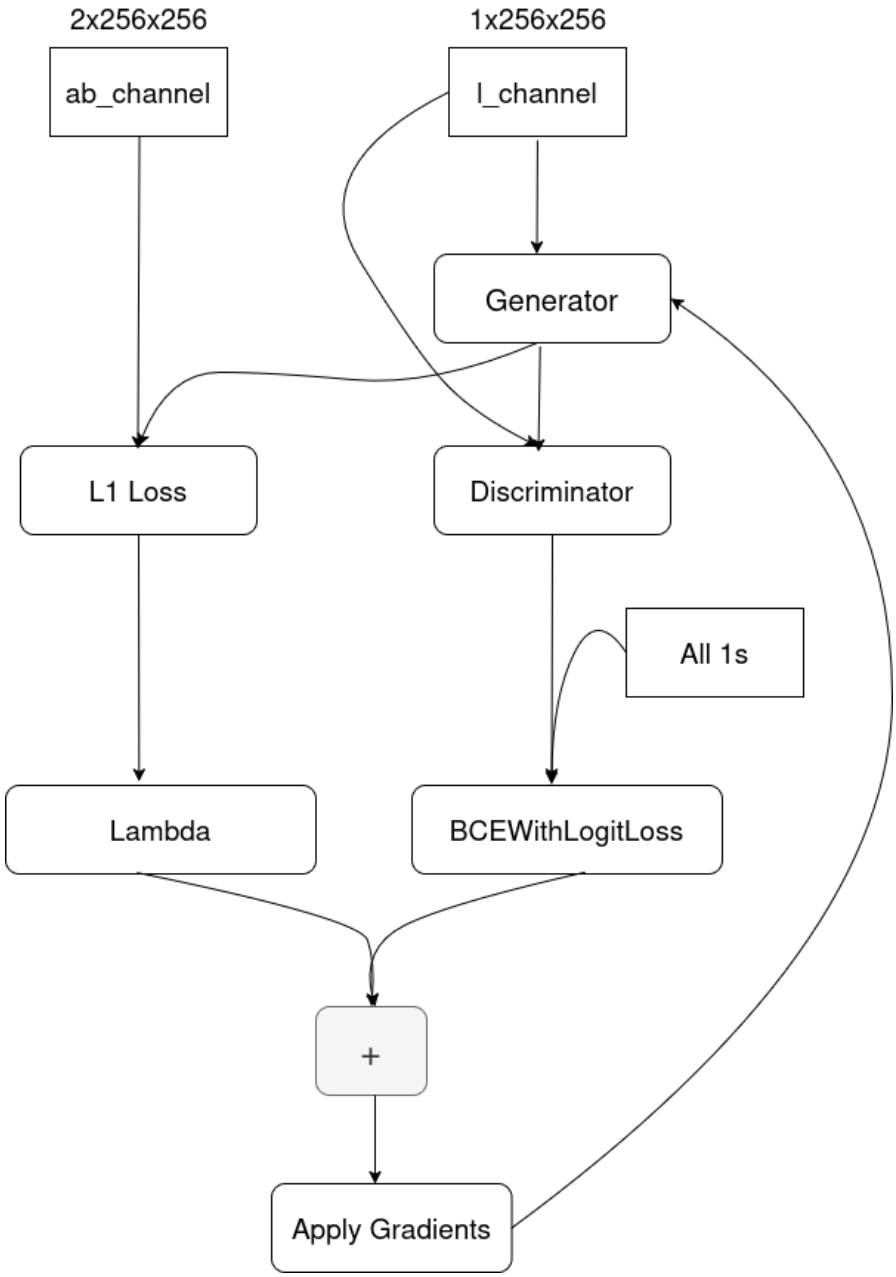

Figure 4.3: Generator training process.

4.6 Evaluation Metrics

Performance was assessed using Peak Signal-to-Noise Ratio (PSNR) on both training and test sets. Training PSNR increased steadily, reflecting effective loss minimization, while test PSNR rose with train PSNR but shows fluctuation, indicating potential overfitting and insufficient regularization/data diversity. Qualitative evaluation involved visual inspection of colorized outputs, focusing on realism and context accuracy (e.g., green grass, blue skies).

In the total GAN loss , which is combination of GAN loss and L1 loss, we can see that validation loss decreasing with training loss but with fluctuation,which is the indicator that our model is quite unstable. At the end we can see that the model is overfitting on training data,which is sign of a unstable model . Though, in general it can colorize images but not everytime with the same perfection.As we know, there is always room for improvement our model too can be improved further.

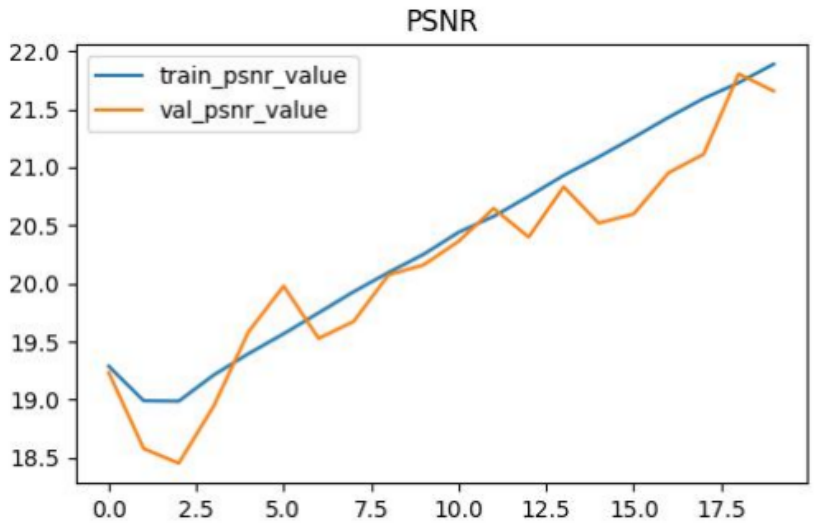

Figure 4.4: PSNR behaviour during training and validation.

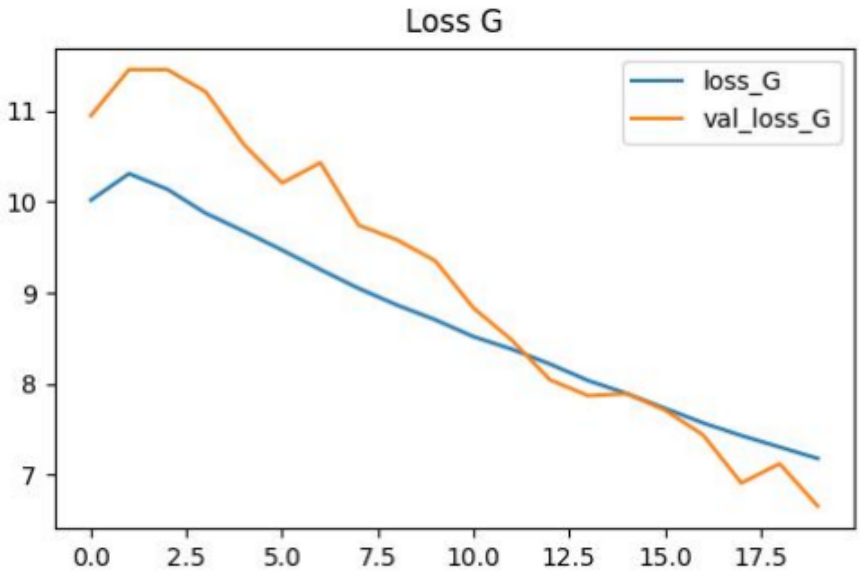

Figure 4.5: Total GAN loss behaviour during training and validation.

4.7 Implementation Details

The model was implemented using Python and PyTorch, leveraging GPU acceleration for efficient training. Key hyperparameters included a batch size of 16 and a 70x70 patch size for the Discriminator. Preprocessing steps, such as normalization, ensured input consistency, while the pretrained ResNet18 weights (from ImageNet) initialized the Generator's encoder, accelerating early convergence.

4.8 Web Application Integration

To enhance usability, a web application was developed to interface with the cGAN colorization model, allowing users to upload grayscale images and receive colorized outputs. This section details the system design.

4.8.1 System Overview

The *System Overview Diagram* (Figure 5.1) illustrates the architecture, comprising a frontend (ReactJS) for user interaction, a backend (NodeJS and Fask) for API handling. It shows data flow from user input to colorized output.

4.8.2 Use Case Diagram

The *Use Case Diagram* (Figure 5.2) depicts user interactions, including uploading grayscale images, viewing colorized results, and downloading images. It highlights the system's functionality for end-users.

4.8.3 Request Response Diagram

The *Request Response Diagram* (Figure 5.3) outlines the data flow, where a user's grayscale image request is processed by the cGAN model via the API, returning a colorized image response. It ensures efficient system operation.

The web app was implemented using NodeJS and Flask for backend integration with the PyTorch-based cGAN, facing challenges like latency during large image processing but demonstrating practical utility for the colorization model.

CHAPTER 5

SYSTEM DESIGN

5.1 System Overview

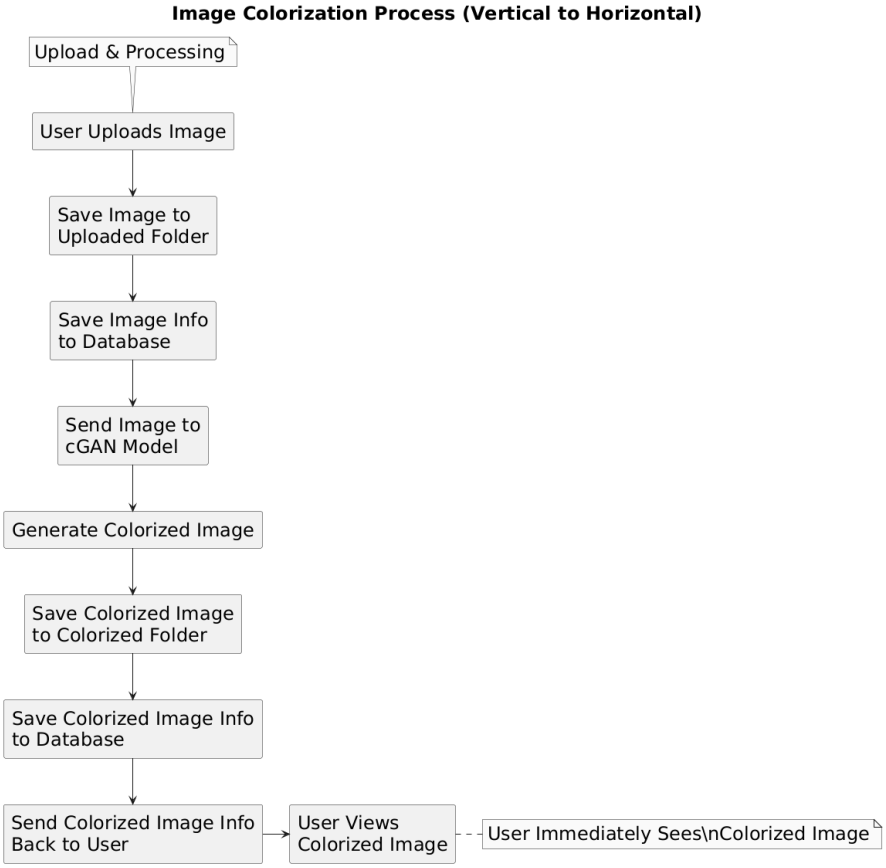

Figure 5.1: system overview.

5.2 Use Case Diagram

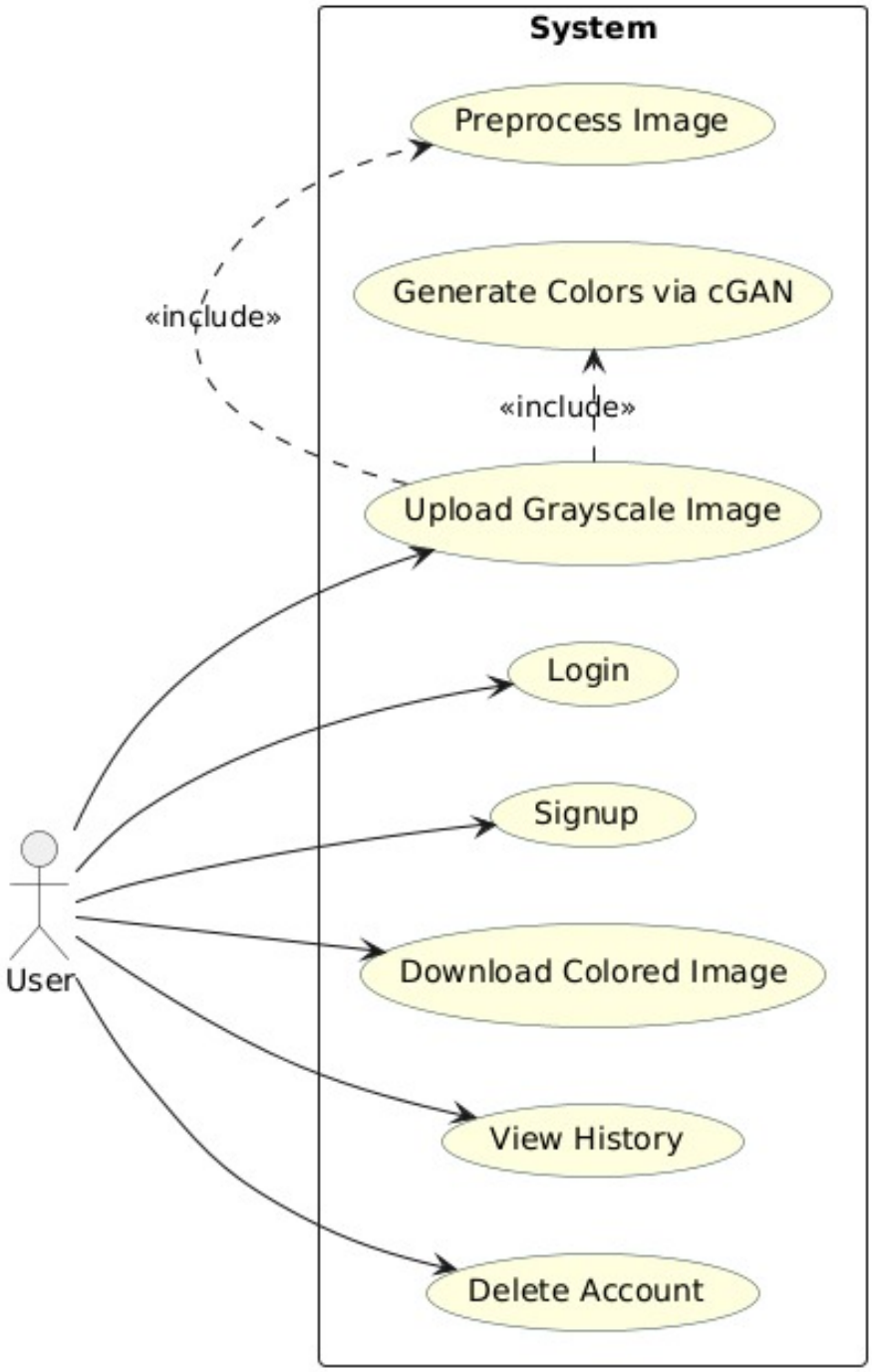

Figure 5.2: Use Case Diagram

5.3 Request Response Diagram

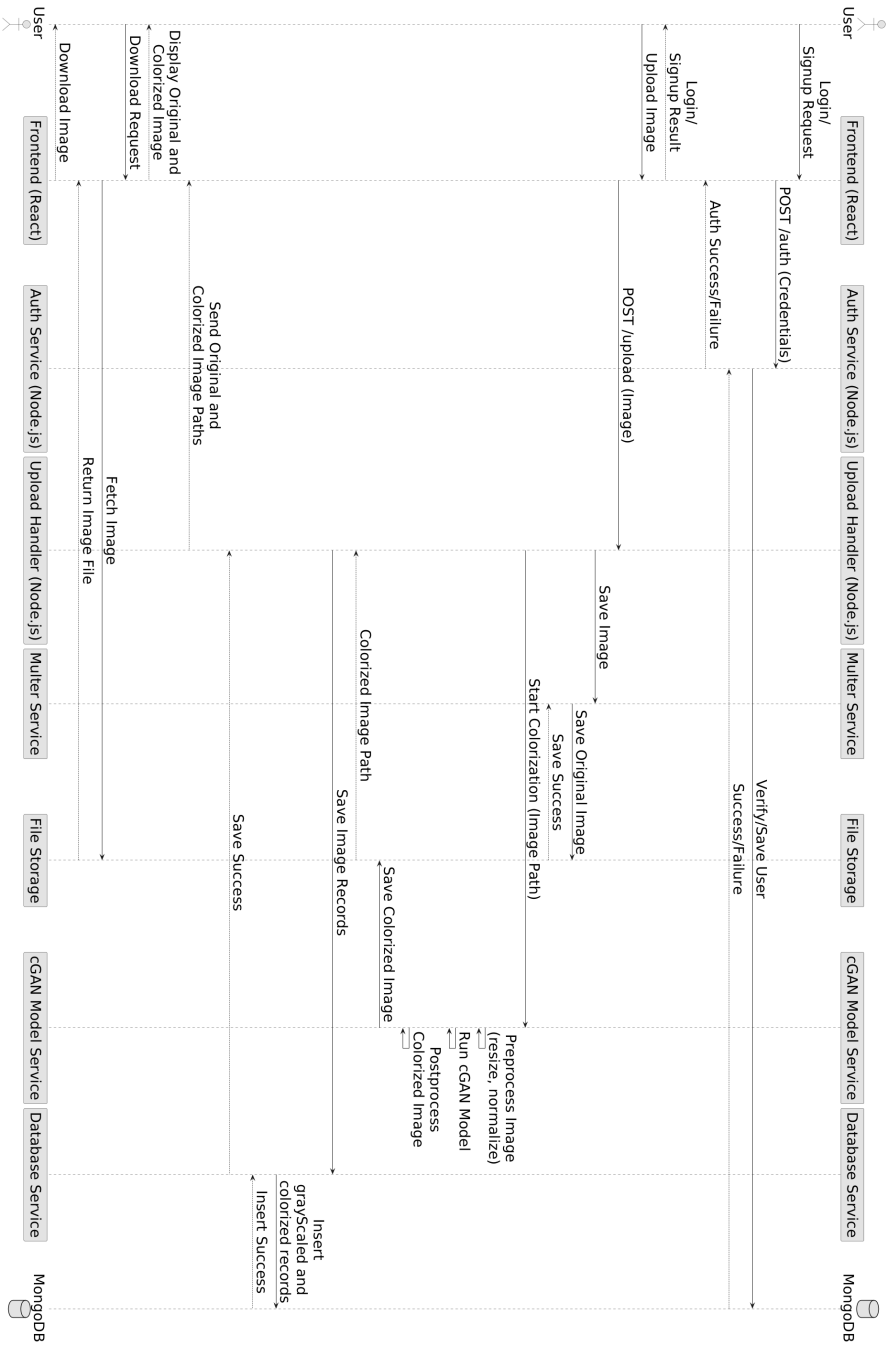

Figure 5.3: Request Response Diagram

CHAPTER 6

RESULT AND CONCLUSION

6.1 Results

Our cGAN-based image colorization model demonstrated promising performance in transforming grayscale images into realistic color outputs. The Generator, utilizing a U-Net architecture with a ResNet18 backbone, and the 70x70 Patch Discriminator successfully produced vibrant and contextually accurate colorizations, such as green landscapes and blue skies. Quantitative evaluation using Peak Signal-to-Noise Ratio (PSNR) showed training PSNR steadily increasing to approximately 34 dB, while test PSNR initially rose to 23 dB but saturated, indicating potential overfitting. Qualitatively, side-by-side comparisons of grayscale inputs and colorized outputs confirmed realism, though minor over-saturation was observed in some cases.

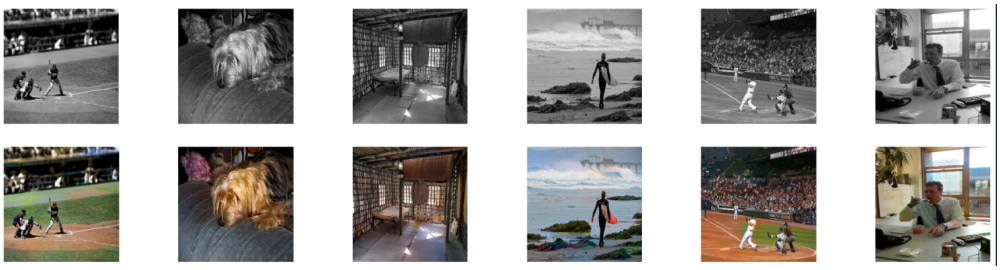

Figure 6.1: Sample grayscale inputs and their colorized outputs using the cGAN model, demonstrating realistic colorization.

6.2 Conclusion

This project successfully implemented a cGAN for automated image colorization, achieving realistic results on training data and reasonable generalization on test data. The model's performance highlights the effectiveness of combining U-Net, ResNet18, and a Patch Discriminator, though challenges like overfitting and color over-saturation suggest areas for improvement. Future work could focus on enhancing generalization through data augmentation, refining loss functions, and scaling the model for high-resolution images to broaden its applicability in historical restoration and digital media.

REFERENCES

[1] Phillip Isola and Jun-Yan Zhu and Tinghui Zhou and Alexei A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks” 2018. [Online]. Available: https://arxiv.org/abs/1611.07004

[2] Richard Zhang and Phillip Isola and Alexei A. Efros, ”Colorful Image Colorization,”2016. [Online]. Available: https://arxiv.org/abs/1603.08511

[3] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” 2018. [Online]. Available: https://arxiv.org/abs/1805.08318

[4] M. Limmer and H. P. A. Lensch, “Infrared colorization using deep convolutional neural networks,” CoRR, vol. abs/1604.02245, 2016. [Online]. Available: http://arxiv.org/abs/1604.02245

[5] A. Deshpande, J. Lu, M. Yeh, and D. A. Forsyth, “Learning diverse image colorization,” CoRR, vol. abs/1612.01958, 2016. [Online]. Available: http://arxiv.org/abs/1612.01958

[6] Kaiming He , Xiangyu Zhang , Shaoqing Ren , Jian Sun, “Deep Residual Learning for Image Recognition,” 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[7] Nazeri, Kamyar and Ng, Eric and Ebrahimi, Mehran, “Image Colorization Using Generative Adversarial Networks,” 2018. [Online]. Available: https://arxiv.org/abs/1803.05400