TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College Of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A MID TERM PROGRESS REPORT
ON
IMAGE COLORIZATION USING GAN

Submitted in partial fulfillment of the requirements for the degree

BACHELOR IN COMPUTER ENGINEERING

Submitted by

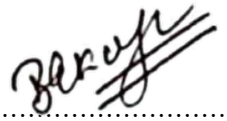
Rabit Khaitu	KCE076BCT028
Romik Gosai	KCE076BCT032
Sagar Suwal	KCE076BCT034
Saman Chauguthi	KCE076BCT035

Under the Supervision of
Er. Niranjan Bekoju
Department of Computer Engineering

Khwopa College Of Engineering
Libali, Bhaktapur
2023-24

Certificate of Approval

The undersigned certify that the final year project entitled **"Image Colorization using GAN"** submitted by Rabit Khaitu, Romik Gosai, Sagar Suwal, Saman Chauguthi to the Department of Computer Engineering in partial fulfillment of requirement for the degree of Bachelor of Engineering in Computer Engineering. The project was carried out under special supervision and within the time frame prescribed by the syllabus.
We found the students to be hardworking, skilled, bona fide and ready to undertake any commercial and industrial work related to their field of study and hence we recommend the award of Bachelor of Computer Engineering degree.



.....
Er. Niranjan Bekoju
Project Supervisor
Machine Learning Engineer
Fusse Machines Nepal Pvt. Ltd.

.....
Er. Dinesh Gothe
Head of Department
Department of Computer Engineering,
Khwopa College of Engineering

Abstract

Image colorization technique is used to colorize the gray-level image or single channel image which is significant and challenging task in image processing. Over the last decade, the process of automatic image colorization have been of significant interest for several application areas including restoration of aged or degraded images. Due to large degrees of freedom during the assignment of color information, this problem is highly ill-posed. Some of the recent development in automatic colorization involve images that contain a common theme or require highly process data such as semantic maps as input. Even though there are various research on image colorization, there are only targeted specifically for the colorization of photographs of old monumentss of Nepal. We proposed to find out best working Generative Adversarial Network (GAN) model for the purpose of colorization of old monuments by comparing different GAN models and colorize old images of antique monuments of Nepal, whose existence is not available or modified today.

Keywords: *Image Colorization, Computer Vision, Generative Adversarial Network*

Contents

List of Tables

List of Figures

List of Symbols and Abbreviation

AI	Artificial Intelligence
API	Application Programming Interface
CIFAR	Canadian Institute For Advanced Research
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
DCGAN	Conditional Deep Convolutional Generative Adversarial Network
GAN	Generative Adversarial Network
HTML	Hypertext Markup Language
MC-GAN	Multidomain Cycle-consistency Generative Adversarial Network
MSE	Mean Squared Error
OpenCV	Open Source Computer Vision Library
PSNR	(Peak Signal-to-Noise Ratio
SAR	Synthetic Aperture Radar
SCGAN	Saliency Map-guided Colorization with Generative Adversarial Network
SSIM	Structural Similarity Index
VAE	Variational Autoencoder
VGG	Visual Geometry Group
VSCode	Virtual Studio Code
UI	User Interface

Chapter 1

Introduction

1.1 Background

Colorization is the process of adding plausible color information to monochrome photographs or videos [1]. Currently, digital colorization of black and white visual data is a crucial task in areas so diverse as advertising and film industries, photography technologies or artist assistance [2]. As colorization requires estimating missing color channels from only one grayscale value, it is inherently an ill-posed problem. Moreover, the plausible solutions of colorization are not unique (e.g., cars in black, blue, or red are all feasible results). Due to the uncertainty and diversity nature of colorization, it remains a challenging task [3].

There are some earliest conventional image colorization methods which can be divided into color transfer-based method (or example-based method) [4] [5] and scribble-based method [6]. In method proposed by Welsh *et al.* [4], colorization was done by matching luminance and texture information between an existing color image and the grayscale image to be colorized. However, this proposed algorithm was defined as a forward problem, thus all solutions were deterministic. The color transfer-based image colorization method proposed by Reinhard *et al.* [5]. The basic idea of this kind of method is to select a color image as reference image, and the color information of the reference image is transmitted into the target gray image that is the high similarity to the reference images. The generalization ability of the transfer-based method is not well, because the process requires a highly similar reference image with the source image. In 2012, Gauge *et al.* [7] proposed an example-based colorization method; this method first analyzed a set of sample color images and extracted the coherent regions of homogeneous textures, and then the grayscale image can be colored by the similar texture of a color image.

In 2014, Goodfellow *et al.* [8] proposed a new type of generative model: generative adversarial networks (GANs). A GAN is composed of two smaller networks called the generator and discriminator. As the name suggests, the generator's task is to produce results that are indistinguishable from real data. The discriminator's task is to classify whether a sample came from the generator's model distribution or the original data distribution. Both of these subnetworks are trained simultaneously until the generator is able to consistently produce results that the discriminator cannot classify.

In [9], a colorization method was proposed by comparing colorization differences between those generated by convolutional neural networks and GAN. The mod-

els in the study not only learn the mapping from input to output image, but also learn a loss function to train this mapping. Their approach was effective in ill-posed problems such as synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images. We aim to extend their approach by generalizing the colorization procedure to high resolution images and suggest training strategies that speed up the process and greatly stabilize it.

1.2 Problem Statement

The Image Colorization project aims to address the conversion of grayscale or black-and-white images into colorized versions that closely resemble the original colored counterparts. By adding realistic and accurate colors to these images, the project seeks to enhance their visual appeal and provide a more immersive experience for viewers. This is particularly important in the context of historical preservation, where colorizing images can bridge the gap between the past and the present, enabling a deeper understanding and appreciation of historical events, places, and people. Additionally, colorization enhances visual quality, conveys important contextual information, promotes accessibility for color-blind individuals, allows for artistic expression, facilitates interactive applications, and serves as a powerful tool for education and documentation. By addressing the challenges associated with image colorization, the project aims to unlock the full potential of grayscale images, enrich visual experiences, preserve historical accuracy, foster understanding and creativity, and have broader implications across various industries.
Even though various Image colorization models are built for different domains, there are very few work done on the domain of traditional monuments of our country. We aim to focus our work on colorizing old monuments photograph whose existence is not available today or there are various modifications done to those monuments and their surroundings.

1.3 Objective

The main aim of this project is:

- To develop a platform that enables conversion of grayscale images of old monuments photographs into colorized version.

Chapter 2

Literature Review

S. Wan, Y. Xia, L. Qi, Y.-H. Yang, and M. Atiquzzaman (2020) [10] introduces an automatic image colorization method using neural networks and optimization. By segmenting grayscale images, extracting features, and propagating color points to neighboring pixels, the proposed method achieves superior colorization results compared to existing algorithms, as demonstrated in experiments on various images.

A. Deshpande, J. Lu, M.-C. Yeh, M. J. Chong, and D. Forsyth (2017) [11] introduces a method for generating multiple diverse colorizations for grayscale images. They use a variational autoencoder (VAE) to learn color field embeddings, incorporate loss terms to prevent blurriness and handle color distribution, and show that their approach outperforms other models in producing diverse colorizations.

B. Li, F. Zhao, Z. Su, X. Liang, Y.-K. Lai, and P. L. Rosin (2017) [12] presents a novel example-based image colorization method that utilizes a sparse representation at the superpixel level. By incorporating various features and a dictionary-based sparse reconstruction approach, the proposed method automatically colorizes grayscale images using a single reference color image. A locality consistent regularization term and an edge-preserving filter are introduced to enhance matching consistency and improve visual quality. Experimental results demonstrate the superiority of the method compared to state-of-the-art techniques in both visual and quantitative evaluations.

F. Ozcelik, U. Alganci, E. Sertel, and G. Unal (2021) [13] proposes a self-supervised learning framework for satellite image pansharpening. Instead of focusing on super-resolution, the approach treats pansharpening as a colorization problem and utilizes a novel PanColorGAN model. By introducing noise injection and adversarial training, the method overcomes spatial-detail loss and blur issues observed in CNN-based pansharpening methods. Experimental results demonstrate the superiority of the proposed approach over previous CNN-based and traditional methods.

G. Ji, Z. Wang, L. Zhou, Y. Xia, S. Zhong, and S. Gong (2021) [14] presents a novel method for colorizing SAR images using a multidomain cycle-consistency GAN (MC-GAN). The approach eliminates the need for paired SAR-optical images and achieves accurate colorization through the use of a mask vector and cycle-consistency loss. Experimental results demonstrate the effectiveness of the proposed method compared to other techniques.

G. Kong, H. Tian, X. Duan, and H. Long (2021) [15] presents a new adversar-

ial edge-aware image colorization method that incorporates semantic information. The proposed method achieves superior results compared to existing approaches by addressing edge color leakage and employing specific loss functions during training. Experimental results validate the effectiveness of the method in image colorization.

Y. Zhao, L.-M. Po, K.-W. Cheung, W.-Y. Yu, and Y. A. U. Rehman (2021) [16] introduces a fully automatic Saliency Map-guided Colorization with Generative Adversarial Network (SCGAN) framework for colorizing grayscale images. The proposed method jointly predicts the colorization and saliency map to minimize semantic confusion and color bleeding. By utilizing pre-trained VGG-16-Gray network and hierarchical discriminators, SCGAN achieves perceptually reasonable colorization with less training data compared to state-of-the-art methods, as demonstrated in evaluations on the ImageNet validation set.

S. Iizuka, E. Simo-Serra, and H. Ishikawa (2016) [17] proposes a deep neural network-based technique for automatic grayscale image colorization that combines global priors and local image features. The network incorporates a fusion layer to merge local information from small image patches with global priors computed using the entire image. The approach is trained end-to-end and can process images of any resolution, achieving significant improvements over the state of the art in terms of realistic and visually appealing colorizations, as validated through a user study and demonstrated across various image types.

K. Nazeri, E. Ng, and M. Ebrahimi (2018) [18] discusses the significance of automatic image colorization and its challenges, particularly in dealing with ill-posed problems and diverse color assignments. The proposed approach focuses on achieving generalization by utilizing a conditional Deep Convolutional Generative Adversarial Network (DCGAN) trained on publicly available datasets. The results of the generative model are compared with traditional deep neural networks, highlighting the effectiveness of the proposed method.

S.N.	**Title**	**Author/s**	**Summary**	**Year**
1	Automated Colorization of a Grayscale Image With Seed Points Propagation [10]	S. Wan, Y. Xia, L. Qi, Y.-H. Yang, and M. Atiquzzaman	uses superpixels, neural network, and optimization to achieve automated colorization with improved accuracy. PSNR:26.18 SSIM:0.83	2020
2	Learning Diverse Image Colorization [11]	A. Deshpande, J. Lu, M.-C. Yeh, M. J. Chong, and D. Forsyth	variational autoencoder (VAE) model with loss terms that address blurry outputs and the uneven distribution of pixel colors, leading to a conditional model	2017

3	Example-Based Image Colorization Using Locality Consistent Sparse Representation [12]	B. Li, F. Zhao, Z. Su, X. Liang, Y.-K. Lai, and P. L. Rosin	example-based image colorization method operating at the superpixel level, utilizing sparse pursuit and a locality consistent sparse representation.	2017
4	Rethinking CNN-Based Pansharpening: Guided Colorization of Panchromatic Images via GANs [13]	F. Ozcelik, U. Alganci, E. Sertel, and G. Unal	PanColorGAN framework using grayscale transformed multispectral images as input, addressing fixed downscale ratio assumptions with noise injection, and employing adversarial training to improve spatial detail.	2021
5	SAR Image Colorization Using Multidomain Cycle-Consistency Generative Adversarial Network [14]	G. Ji, Z. Wang, L. Zhou, Y. Xia, S. Zhong, and S. Gong	colorizing synthetic aperture radar (SAR) images using a multidomain cycle-consistency generative adversarial network MC-GAN, which incorporates a mask vector and multidomain classification loss. PSNR:14.066 SSIM:0.117	2021
6	Adversarial Edge-Aware Image Colorization With Semantic Segmentation [15]	G. Kong, H. Tian, X. Duan, and H. Long	combines adversarial edge-aware image colorization with semantic segmentation, using a deep semantic fusion generator to infer semantic clues. PSNR:31.359 SSIM:0.972	2021
7	SCGAN: Saliency Map-Guided Colorization With Generative Adversarial Network [16]	Y. Zhao, L.-M. Po, K.-W. Cheung, W.-Y. Yu, and Y. A. U. Rehman	framework that combines color prediction with saliency map guidance to minimize semantic confusion and color bleeding. PSNR:23.80 SSIM:0.9473	2021
8	joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification [17]	S. Iizuka, E. Simo-Serra, and H. Ishikawa	deep learning-based approach that combines global priors and local image features. PSNR:29.851 SSIM:0.955	2016

9	Image colorization using generative adversarial networks [18]	K. Nazeri, E. Ng, and M. Ebrahimi	conditional Deep Convolutional Generative Adversarial Network (DCGAN) trained on publicly available datasets to fully generalize the image colorization process. PSNR:29.415 SSIM:0.928	2018

Table 2.1: Review Matrix with Research Papers, Authors, Summary & Year of Publication.

Chapter 3

System (or Project) Design and Architecture

We aim to develop a system that accepts grayscale images as input and transforms them into colourful images. The system will utilize various algorithms and techniques to generate images that closely resemble realistic color of image, capturing the details, colors and textures of the subject portrayed in the input grayscale image. This tool will enable to visualize old black and white images into a colourful form.

3.1 Use case Diagram

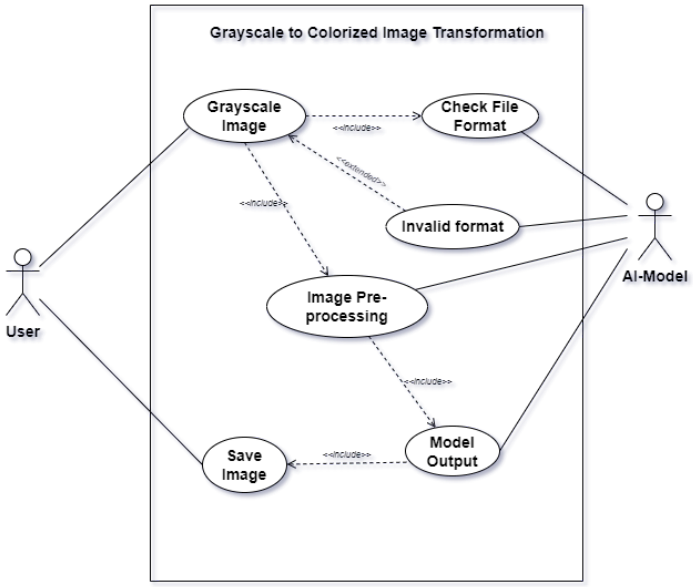


Figure 3.1: Use Case Diagram

The user can input grayscale image in the UI, and that image will undergo preprocessing before being passed to an AI model for output generation. The AI model will generate an output based on the grayscale image. Finally, the user can save the final result.

3.2 System Block Diagram

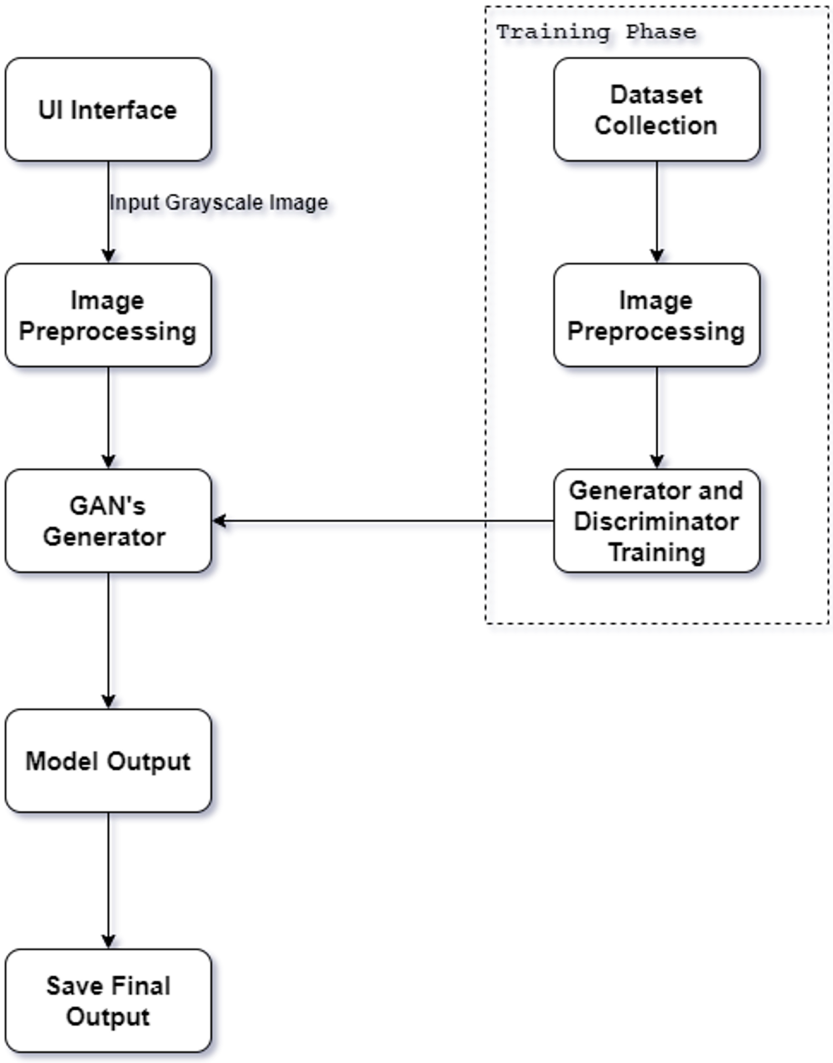


Figure 3.2: System Block Diagram

The user begins by uploading an grayscale image on the UI interface. The image undergoes preprocessing before being fed into an AI model. The model generates an output image based on the input grayscale image. Finally, the user has the option to save the transformed image as the final result for further use or sharing.

Chapter 4

Methodology

4.1 Software Development Approach

Prototype model is a software development model where instead of freezing the requirements before design or coding can proceed, a throwaway prototype is built to understand the requirements. The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality. In this model, we create the prototype of the actual system, update the requirements and again rebuild the system until the final requirements are met.

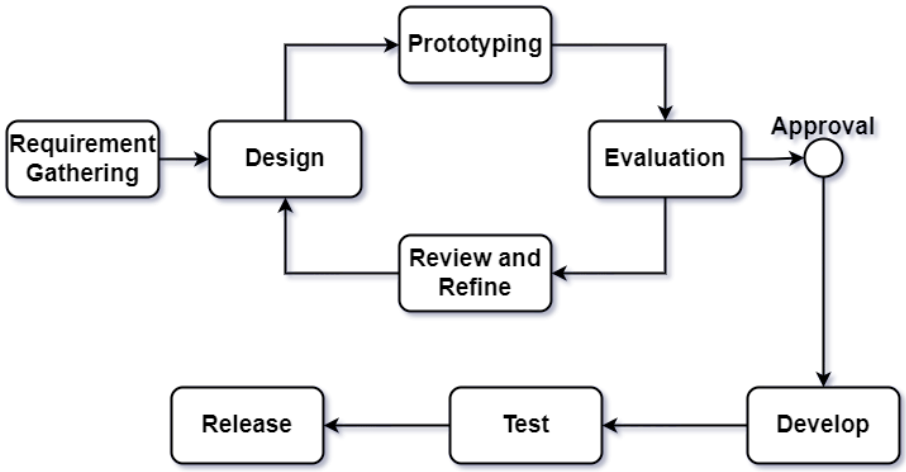


Figure 4.1: Prototype Model for Software Development

The different phases of the prototyping model as are:

4.1.1 Requirements gathering and analysis

A prototyping model starts with requirement analysis. In this phase, the requirements of the system are defined in detail. During the process, the users of the system are interviewed to know what is their expectations from the system. In this phase, business analysts and other individuals responsible for collecting the requirements and discussing the need for the product, meet the stakeholders or clients.

4.1.2 Quick design

The second phase is a preliminary design or a quick design. In this stage, a simple design of the system is created. However, it is not a complete design. It gives a brief idea of the system to the user. The quick design helps in developing the prototype.

4.1.3 Build a Prototype

In this phase, an actual prototype is designed based on the information gathered from a quick design. It is a small working model of the required system.

4.1.4 Initial user evaluation

In this stage, the proposed system is presented to the client for an initial evaluation. In this phase, the initial prototype is deployed and is accessible to clients for its use. Clients review or evaluate the prototype and they provide their feedback to the requirements gathering and development teams. It helps to find out the strength and weaknesses of the working model. Comment and suggestions are collected from the customer and provided to the developer.

4.1.5 Refining prototype

If the user is not happy with the current prototype, you need to refine the prototype according to the user's feedback and suggestions. This phase will not over until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed based on the approved final prototype.

4.1.6 Implement Product and Maintain

Once the final system is developed based on the final prototype, it is thoroughly tested and deployed to production. The system undergoes routine maintenance for minimizing downtime and prevent large-scale failures.

4.2 Description of Work Flow

4.2.1 Data Collection

The process of collecting data for AI training involves several key steps. First, the objective of the project is defined. Then, relevant data sources are identified, and the data is gathered from these sources. The collected data is preprocessed to ensure its quality and relevance for the task at hand. Annotation or labeling may be applied to the data to provide necessary information for training. If needed, the dataset is augmented to increase its size or diversity.
Once the dataset is ready, it is split into training, validation, and testing sets to assess the performance of the AI model. Throughout this process, privacy regulations and ethical considerations must be strictly adhered to, ensuring that sensitive

information is protected. Maintaining data quality is crucial, and documentation of the collected data is important for efficient training and future reference.
In this specific project, the datasets will be sourced from popular websites such as Kaggle.com, as well as specific datasets like CIFAR-10 and places365, among other relevant sources.

4.2.2 Data Preprocessing

Image data preprocessing is a crucial step in preparing images for AI model training or inference. It involves several steps to ensure that the images are in a suitable format for the model to process effectively.

4.2.2.1 Resizing

The first step is resizing the images to a standard resolution. This is done to ensure consistency in image dimensions, as models typically expect inputs of a fixed size. Resizing also helps to reduce computational requirements during training or inference.

4.2.2.2 Normalization

The next step is normalizing pixel values. This involves transforming the pixel values to a common scale, such as between 0 and 1 or -1 and 1. Normalization helps to improve model convergence and stability by reducing the impact of varying pixel value ranges across different images.

4.2.2.3 Data Augmentation

Data augmentation techniques are then applied to increase the diversity of the dataset. This involves applying transformations such as rotation, scaling, flipping, or cropping to the images. Data augmentation helps in reducing overfitting and improving the model's ability to generalize by exposing it to variations and deformations present in real-world data.

4.2.3 Model Selection

For our project, we have chosen to implement few powerful and versatile deep learning models with Generative Adversarial Network(GAN) architectures such as MC-GAN, DCGAN, conditional GAN, SCGAN and find the best one through evaluations. Generative Adversarial Networks refer to a family of generative models that seek to discover the underlying distribution behind a certain data generating process. This distribution is discovered through an adversarial competition between a generator (G) and a discriminator(D). In GAN, the two models are trained such that the discriminator strives to distinguish between generated and true examples, while the generator seeks to confuse the discriminator by producing data that are as realistic and compelling as possible. Loss function of GAN is given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_x[log\ D(x)] + \mathbb{E}_z[log(1 - D(G(z)))] \tag{4.1}$$

where $G(z), D(x), D(G(z))$ denote the generator's output when it receives as input noise z, the discriminator's probability that the original data x is real, and the discriminator's probability that a synthetic sample $G(z)$ of data is real, and $\mathbb{E}_x, \mathbb{E}_z$ denote the mean likelihood over all original data and synthetic data respectively. In the above equation, we notice that during the training of the discriminator (D) focuses on maximizing the $log\ D(x)$, which means achieving the correct label (real or synthetic) during the classification of x, while, the generator (G) focuses on minimizing $log(1 - D(G(z)))$ and cannot directly influence $logD(x)$.

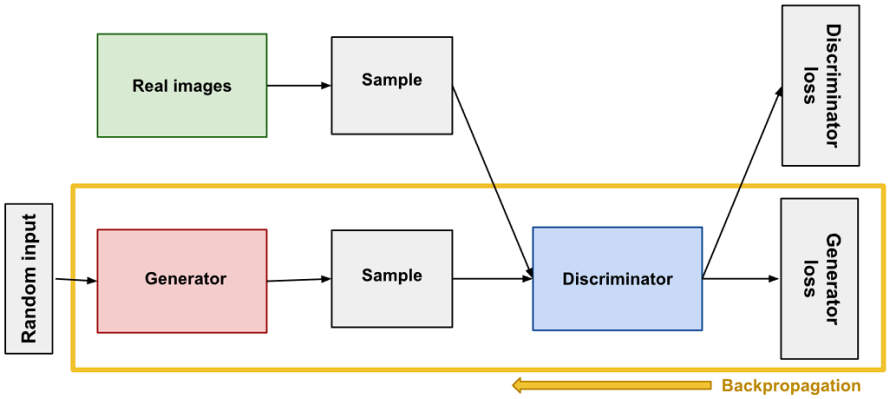


Figure 4.2: Architecture of GAN

4.2.3.1 Generator

The generator network is responsible for generating synthetic data, such as images, texts, or any other desired output. It takes random noise as input and produces synthetic samples that resemble the real data. The generator aims to learn the underlying patterns and distribution of the training data, enabling it to generate realistic samples that are difficult to distinguish from the real ones.
The generator typically consists of one or more layers of neural networks, such as fully connected layers or convolutional layers. The input noise is fed through these layers, gradually transforming it into a meaningful output that mimics the real data. The generator is trained to minimize the differences between the generated samples and the real samples.

4.2.3.2 Discriminator

The discriminator network acts as a binary classifier that distinguishes between real and generated data. It takes an input sample, either real or synthetic, and predicts whether it belongs to the real data distribution or the generated data distribution. The discriminator is trained to become increasingly accurate in its classification task.
Similar to the generator, the discriminator is typically composed of neural network layers. It takes the input sample and passes it through the layers to produce a probability score indicating the likelihood of the sample being real or fake. The discriminator is trained to maximize the probability of correctly classifying real and generated samples.

4.2.3.3 Training Process

The generator and discriminator are trained together in a competitive manner. The process can be summarized as follows:

1. The generator generates synthetic samples using random noise as input.
2. The discriminator receives both real and generated samples and tries to classify them correctly.
3. The discriminator's performance is measured using a loss function (e.g., binary cross-entropy), indicating how well it distinguishes real and generated samples.
4. The generator's performance is measured based on the discriminator's output when fed with generated samples.
5. The generator is updated to improve its ability to generate samples that deceive the discriminator.
6. The discriminator is updated to improve its ability to correctly classify real and generated samples.
7. Steps 1-6 are repeated iteratively, allowing the generator and discriminator to learn from each other and improve their performance.

The objective of training the GAN is for the generator to become better at generating realistic samples that can fool the discriminator, while the discriminator aims to become better at accurately classifying real and generated samples. As the training progresses, both the generator and discriminator tend to improve, ultimately leading to the generation of high-quality synthetic samples.

4.2.4 Model Evaluation

We will assess the performance of our implemented models. We will utilize objective metrics such as PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index) for evaluating the quality of colorized images. They provide quantitative measures to assess the similarity between the original and colorized images.

4.2.4.1 PSNR(Peak Signal-to-Noise Ratio)

PSNR is a metric used to quantify the amount of noise or distortion between two images. It measures the ratio between the peak signal power and the mean squared error (MSE) between the original and colorized images. The higher the PSNR value, the closer the colorized image is to the original image in terms of quality. PSNR primarily focuses on pixel-wise differences and does not consider human perception. PSNR is calculated using the following formula:

$$PSNR = 20log_{10}(MAX_I) - 10log_{10}(MSE) \tag{4.2}$$

where,
MAX_I is the maximum pixel value (e.g., 255 for an 8-bit image).

MSE is the mean squared error between the original and colorized images. PSNR is typically expressed in decibels (dB), and higher PSNR values indicate better quality.

4.2.4.2 SSIM (Structural Similarity Index)

SSIM is a metric that measures the structural similarity between two images. It takes into account three components: luminance, contrast, and structural similarity. Unlike PSNR, which only considers pixel-wise differences, SSIM also incorporates perceptual information to assess the visual quality of the colorized image. SSIM is calculated by comparing the luminance, contrast, and structure of the original and colorized images. The SSIM index ranges from -1 to 1, where 1 indicates a perfect match between the images.

4.2.5 Model Deployment

After optimizing the model based on continuous evaluation and feedback, we proceed with deploying the model and creating the UI interface, we will utilize Flutter and Flask. Flask will be used to deploy the model by creating a server that handles API requests and hosts the model. This server will process image inputs from users and colorize accordingly. Flutter will be employed to create an intuitive user interface with features such as image file upload, real-time image capture, and ressult of colorized image. Communication between the Flutter frontend and Flask backend will be established through API requests. The combined power of Flutter and Flask will enable users to interact with the application, convert grayscale image into colorized form.

Chapter 5

Work Completed

5.1 Data Gathering

Our image colorization project depends on large number of image datasets of old monuments that includes various temples, stupas, landmarks etc. We've successfully collected 1396 images of old monuments which are in colorized form that can be used for training and testing of the proposed model. Out of 1396 images, 200 of them are seperated for testing purposes.
We also collected few old images of monuments whose existence are only available in grayscale image. We aim to build a working model that can realistically colorized these old images of various monuments.

5.2 Data Preprocessing

Pre-processing is required in order for the collected images to fit the model building properly and enhance the outcome of the model. We have performed following steps in order for the image preprocessing in the preparation of dataset.

5.2.1 Resize

For the model building and training process, we decided to take images of size 256x256. It is used for both training input and target. So we converted all collected images into size of 256x256.

5.2.2 RGB to Grayscale Conversion

For the model, we need grayscale input and output as RGB image. Since we collected RGB images, we needed to convert these RGB images into grayscale that act as an input for the proposed model. In conversion process we took various ratios of 3 channels i.e. RGB and set values of all channels same. The value were taken as $0.2989\alpha_R + 0.5870\alpha_G + 0.1140\alpha_B$
where $\alpha_R, \alpha_G and \alpha_B$ are values of Red, Green and Blue channels of the particular pixels.

5.2.3 Normalization

All the values of input image are converted into the range of -1 to 1 by mapping value of 0 in pixel to -1 and 255 to 1. This conversion is done by following equation.

$$x_{normalized} = \frac{x}{127.5} - 1 \tag{5.1}$$

5.3 Prepare Model

To generate colorized image from the input grayscale image, we used image to image translation GAN architecture i.e. pix2pix cGAN is mainly used to generate some images from equivalent image as input which is translated through encoder and decoder during the process. This GAN contained 2 main components which are Generator and Discriminator.

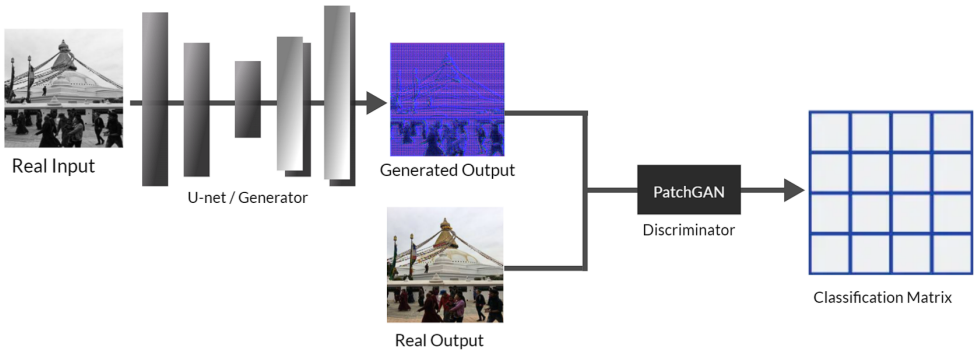


Figure 5.1: Pix2pix Model

5.3.1 Generator

The generator in the Pix2Pix cGAN model consists of an encoder-decoder architecture. The encoder takes the grayscale input image and progressively reduces its spatial dimensions while increasing its feature depth. This encoded representation is then passed to the decoder, which upscales and transforms it to generate the final colored image. Skip connections are typically used to connect corresponding encoder and decoder layers, helping to preserve fine details.
The generator's goal is to create colorized images that are convincing and indistinguishable from real colored images. To achieve this, it's trained alongside a discriminator, which is a separate neural network. The discriminator's job is to differentiate between real colored images and those produced by the generator. The generator aims to improve its performance by trying to generate images that can fool the discriminator into believing they are real.

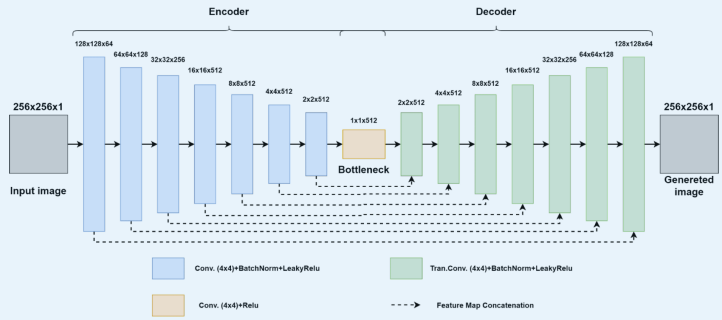


Figure 5.2: U-net Generator in pix2pix

5.3.2 Discriminator

The PatchGAN discriminator is a key component of the Pix2Pix model used for image-to-image translation tasks like image colorization. Unlike traditional discriminators that classify the entire input image as real or fake, the PatchGAN discriminator focuses on making judgments at the patch level. This allows it to provide fine-grained feedback on the realism of different parts of the generated image. PatchGAN discriminator works by sliding a small window (patch) over both the real target images and the images generated by the generator. For each patch, the discriminator produces a probability score indicating whether the patch is real or fake. These probability scores are used to compute a loss that guides the training process.

The advantage of using a PatchGAN discriminator lies in its ability to provide spatially detailed feedback. Instead of generating a single probability for the entire image, it generates a grid of probabilities, effectively allowing the model to capture local details and textures. This is particularly useful for tasks like image colorization, where the quality of colorization can vary significantly across different regions of an image.

The PatchGAN discriminator's outputs can be thought of as a heatmap of probabilities, indicating the likelihood that each patch in the input image is real or fake.

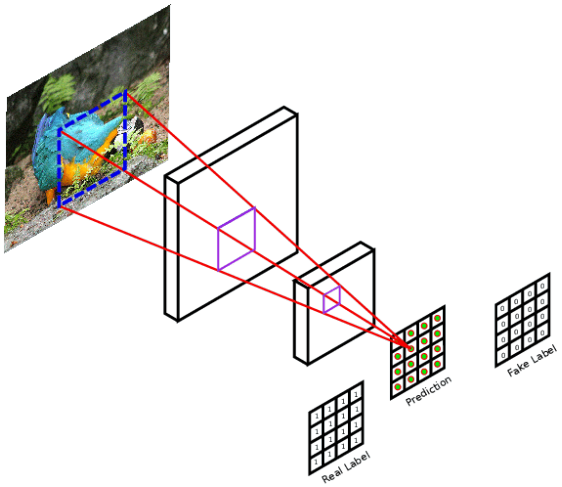


Figure 5.3: PatchGAN Discriminator in pix2pix

5.4 Model Evaluation

After the model is defined and trained, the model is evaluated based on different generator and discriminator losses.

5.4.1 Discriminator loss

Discriminator loss is typically computed using a binary cross-entropy loss function. This loss function measures the difference between the predicted probabilities by the discriminator for real and generated images and the actual labels. This also had 2 components which are:

5.4.1.1 Discriminator Loss for Real Samples

The discriminator's task is to correctly classify real samples (images from the true dataset) as real. The loss for real samples is calculated based on how well the discriminator's predictions match the target labels (which are all ones, indicating real samples).Mathematically, the loss for real samples is given by:

$$L_{real} = -\frac{1}{N}\sum_{i=1}^{N} logD(x_i) \tag{5.2}$$

where, N is the number of samples in current batch.
x_i representss the i-th real sample
$D(x_i)$ is the discriminator's predicted probability that x_i is a real sample.

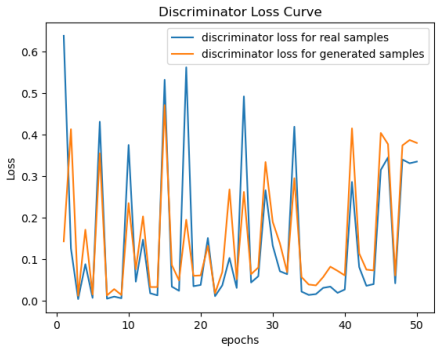
5.4.1.2 Discriminator's Loss for Generated Samples

The discriminator's second task is to correctly classify generated samples (images produced by the generator) as fake. The loss for generated samples is calculated based on how well the discriminator's predictions match the target labels. Mathematically, the loss for generated samples is given by:

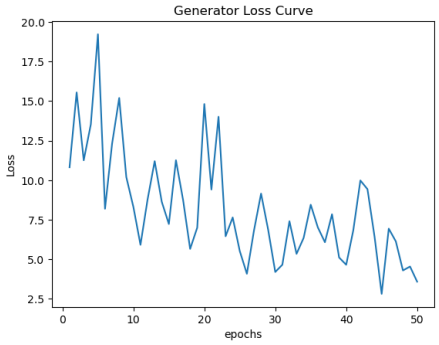
$$L_{generated} = -\frac{1}{N}\sum_{i=1}^{N} log(1 - D(G(z_i))) \tag{5.3}$$

where, N is the number of samples in current batch.
z_i is the number of samples in current batch.
$G(z_i$ is the i-th generated sample produced by the generator using z_i
$D(G(z_i))$ is the discriminator's predicted probability that $G(z_i$ is a real sample (which the generator aims to make as low as possible).

$$L_{discriminator} = L_{real} + L_{generated} \tag{5.4}$$



(a) Discriminator Loss



(b) Generator Loss

5.4.2 Generator Loss

the generator's primary objective is to produce images that are indistinguishable from real images to the discriminator. The generator loss is a key part of achieving this goal. The loss guides the generator's learning process by quantifying how well it is able to fool the discriminator. The loss typically involves two parts:

5.4.2.1 Adversarial Loss

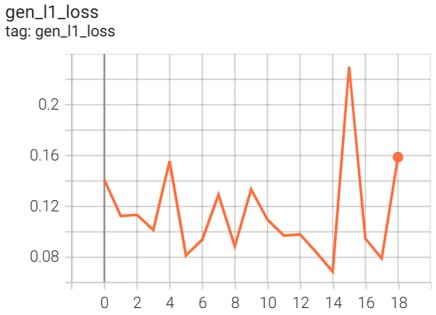
The adversarial loss, often referred to as the "generator adversarial loss," is based on how well the generator can deceive the discriminator. It is calculated using the predicted probabilities of the discriminator for the generated images. The generator aims to minimize this loss by producing images that the discriminator cannot confidently classify as fake. Mathematically, the adversarial loss is given by:

$$L_{adv} = -\frac{1}{N}\sum_{i=1}^{N} log(D(G(z_i))) \tag{5.5}$$

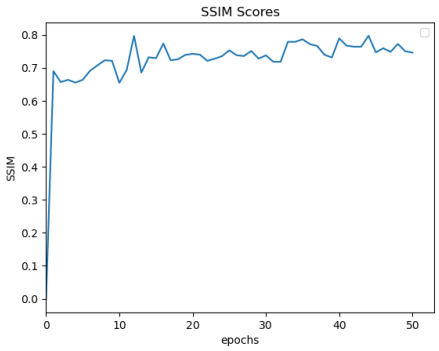
where, N is the number of samples in current batch.
z_i is the number of samples in current batch.
$G(z_i$ is the i-th generated sample produced by the generator using z_i
$D(G(z_i))$ is the discriminator's predicted probability that $G(z_i$ is a real sample (which the generator aims to make as low as possible).

5.4.2.2 L1 loss

The L1 loss, also known as the mean absolute error (MAE) loss, is a commonly used loss function in machine learning and deep learning tasks, including image-to-image translation models like Pix2Pix. It measures the average absolute difference between the predicted values and the ground truth values. In the context of Pix2Pix and other image translation tasks, L1 loss is often used to ensure that the generated images closely match the target images. L1 loss encourages the generator to produce images that closely match the ground truth images in terms of pixel values. This helps ensure that the generated colorized images exhibit accurate colors and details, promoting realistic and visually pleasing results. Mathematically,



(a) Generator L1 loss



(b) SSIM Evaluation

the L1 loss between two sets of data points y_i and $\hat{y}_i$ is calculated as:

$$L_{L1} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{5.6}$$

where, N is the number of data points (usually pixels) in the images.
y_i represents the ground truth value at data point i.
$\hat{y}_i$ represents the predicted value at data point i (in the context of Pix2Pix, this would be the value generated by the model).

$$L_{generator} = L_{adv} + \lambda . L_{L1} \tag{5.7}$$

Here, λ is a hyperparameter that balances the importance of the adversarial loss and the additional losses. It determines how much emphasis is placed on fooling the discriminator versus other objectives. For our model we took $\lambda = 100$

5.4.3 SSIM evaluation

SSIM index produces a value between -1 and 1, where 1 indicates that the two images are identical, and a higher value generally implies better similarity. We calculated SSIM score for one image after each epoch and the result was as in 5.5b.

5.5 Run Test Samples

We also run trained model to generated colorized image for some old images of monuments. And results were as follows:

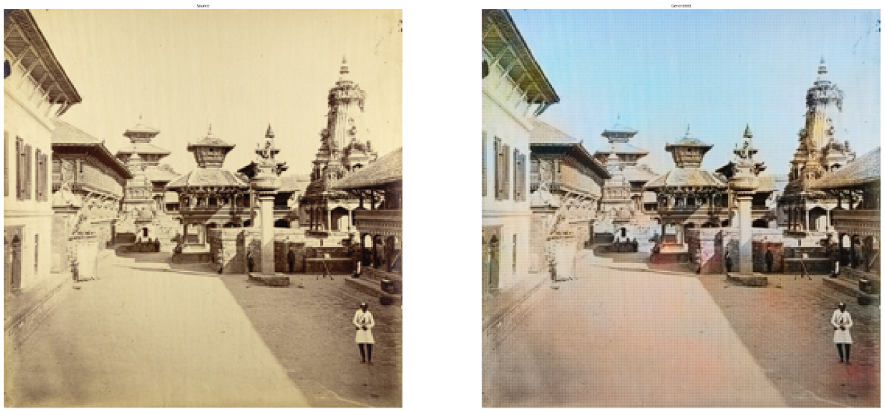


Figure 5.6: test image 1

Figure 5.7: test image 2

Figure 5.8: test image 3

Chapter 6

Work in Progress

6.1 Improvement in Dataset

With limitation of time and resource we haven't been able to collect enough amount of datasets required for good generation of colorized image. We will collect more datasets for the good model training

6.2 Model Refinement

The model we built is not able to produce enough accurate result. We will have to study more on how to increase the accuracy of the model and refine it for the better model building.

6.3 Model Evaluation on Benchmark Dataset

We haven't yet tested this model on benchmark datasets for image colorization. It will help us compare our models to present state of art models and help in improving the model.

6.4 Model Deployment

The deployment of the model is going to be on web where user can upload grayscale image and see colorized output image which they can download as well.

Chapter 7

Conclusion

In conclusion, our mid-term defense report showcases the progress we've made in building a prototype of our model for the project. While we have successfully developed the prototype, it's important to note that the optimization phase is yet to be initiated. We acknowledge that we are slightly behind schedule. However, our team is committed to bridging this gap by dedicating extra effort during the upcoming holiday period. We are confident that with our collective determination and hard work, we will not only catch up but also propel the project towards successful completion. This mid-term assessment serves as a valuable checkpoint, motivating us to stay focused and devoted to achieving our goals.

Bibliography

[1] L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," *IEEE transactions on image processing*, vol. 15, no. 5, pp. 1120–1129, 2006.

[2] P. Vitoria, L. Raad, and C. Ballester, "Chromagan: Adversarial picture colorization with semantic class distribution," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2445–2454.

[3] Y. Wu, X. Wang, Y. Li, H. Zhang, X. Zhao, and Y. Shan, "Towards vivid and diverse image colorization with generative color prior," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 14 377–14 386.

[4] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 277–280.

[5] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer graphics and applications*, vol. 21, no. 5, pp. 34–41, 2001.

[6] G. Zong, Y. Chen, G. Cao, and J. Dong, "Fast image colorization based on local and global consistency," in *2015 8th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1. IEEE, 2015, pp. 366–369.

[7] C. Gauge and S. Sasi, "Automated colorization of grayscale images using texture descriptors and a modified fuzzy c-means clustering," 2012.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets (advances in neural information processing systems)(pp. 2672–2680)," *Red Hook, NY Curran*, 2014.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[10] S. Wan, Y. Xia, L. Qi, Y.-H. Yang, and M. Atiquzzaman, "Automated colorization of a grayscale image with seed points propagation," *IEEE Transactions on Multimedia*, vol. 22, no. 7, pp. 1756–1768, 2020.

[11] A. Deshpande, J. Lu, M.-C. Yeh, M. J. Chong, and D. Forsyth, "Learning diverse image colorization," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2877–2885.

[12] B. Li, F. Zhao, Z. Su, X. Liang, Y.-K. Lai, and P. L. Rosin, "Example-based image colorization using locality consistent sparse representation," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5188–5202, 2017.

[13] F. Ozcelik, U. Alganci, E. Sertel, and G. Unal, "Rethinking cnn-based pansharpening: Guided colorization of panchromatic images via gans," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 4, pp. 3486–3501, 2021.

[14] G. Ji, Z. Wang, L. Zhou, Y. Xia, S. Zhong, and S. Gong, "Sar image colorization using multidomain cycle-consistency generative adversarial network," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 2, pp. 296–300, 2021.

[15] G. Kong, H. Tian, X. Duan, and H. Long, "Adversarial edge-aware image colorization with semantic segmentation," *IEEE Access*, vol. 9, pp. 28 194–28 203, 2021.

[16] Y. Zhao, L.-M. Po, K.-W. Cheung, W.-Y. Yu, and Y. A. U. Rehman, "Scgan: Saliency map-guided colorization with generative adversarial network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 8, pp. 3062–3077, 2021.

[17] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 4, pp. 1–11, 2016.

[18] K. Nazeri, E. Ng, and M. Ebrahimi, "Image colorization using generative adversarial networks," in *Articulated Motion and Deformable Objects: 10th International Conference, AMDO 2018, Palma de Mallorca, Spain, July 12-13, 2018, Proceedings 10*. Springer, 2018, pp. 85–94.