Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A project report on

"Shades of History: Reviving Nepal's Heritage through AI"

[Code No. AISP 311]

Submitted By:

Shlok Koirala (17)

Anshu Patel (21)

Nitesh Rajbanshi (25)

Submitted To:

Department of Computer Science and Engineering

June 18, 2023

Panchkhal, Kavre

Acknowledgement

The completion of this report, "Shades of History: Reviving Nepal's Heritage through AI", would not have been possible without the invaluable support of several individuals and institutions.

We are grateful to the Department of Computer Science and Engineering (DoCSE), School of Engineering (SOE), and Kathmandu University (KU) for providing the academic foundation and resources necessary for this project. Their commitment to fostering a stimulating learning environment is deeply appreciated.

We are particularly grateful to our esteemed supervisor, Dr. Yagya Yaj Pandeya. His guidance, encouragement, and expertise were instrumental throughout the research and development process. His insights and constructive feedback significantly enhanced the quality of this work.

Lastly, we would like to acknowledge the invaluable contributions of all those who have directly or indirectly contributed to this project.

Yours Sincerely,

Shlok Koirala

Anshu Patel

Nitesh Rajbanshi

Abstract

This project explores the enhancement and restoration of historical black and white photographs of Nepal using advanced deep learning techniques. The primary objective is to colorize and outpaint these images, providing a vivid and extended visual representation of Nepal's heritage. For the colorization task, models such as Generative Adversarial Networks (GANs) and Deeplab are employed. A novel training strategy is introduced to address the initial challenge of both the generator and discriminator lacking task knowledge. This involves pretraining the generator in a supervised manner using a U-Net with a pretrained ResNet18 on ImageNet, initially with L1 loss on the colorization task, followed by a combination of adversarial and L1 loss for further refinement. Additionally, for the outpainting task, PQDiffusion is utilized to upscale images threefold and predict the extended surrounding areas, generating comprehensive outprinted images. This dual approach not only revitalizes historical images with accurate colors but also extends their visual context, preserving and enhancing Nepal's cultural heritage for future generations.

Keywords: Image Outpainting, Generative Adversarial Networks (GANs), Diffusion Models, Vector Quantized Variational Autoencoder (VQ-VAE), Vision Transformer (ViT), Positional Query Diffusion Model (PQDiff), Image Colorization, Conditional GANs (cGANs), Deep Learning, Data Preprocessing

Table of Contents

Acknowledgement i

Abstract ii

List of Figures v

List of Tables vi

Acronyms vii

Chapter 1: Introduction 1

1.1 Background 1

1.2 Objectives 1

1.3 Problem Statement 1

1.4 Motivation and Significance 2

Chapter 2: Related Works 3

Chapter 3: Methodology 4

3.1 Image Colorization 4

3.2 Data Collection 4

3.3 Data Preprocessing 4

3.4 Generative Adversarial Network 6

3.5 Objective Function 7

3.5.1 Discriminator Backward Propagation 7

3.5.2 Generator backpropagation 8

3.6 UNET 8

3.6.1 UNET Architecture Description 8

3.6.2 New Generator Architecture and Pretraining Strategy 10

3.6.3 Pretraining Strategy: 10

3.6.4 Integration into GAN Framework: 11

3.7 PatchGAN (Markovian Discriminator) 11

3.7.1 Discriminator Architecture Description 11

3.8 Deeplabv3 Generator 12

3.8.1 Detailed DeepLabv3 Generator Architecture 12

3.9 Image Outpainting 13

3.10 Data Preprocessing 14

3.11 Vector Quantized Variational Autoencoder 14

3.12 Diffusion 15

3.13 Vision Transformer 15
3.14 Positional query diffusion model (PQDiff's) 16
3.14.1 Relative Positional Embedding 16
3.14.2 Forward and Backward of Diffusion 18
3.15 The Positional-Aware Transformer Model $\boldsymbol{g\theta}$ 19
3.16 Sampling Pipeline 20
3.17 Training Details 20
3.18 Evaluation and Baselines 21
3.18.1 Fréchet Inception Distance 21
3.18.2 Inception Score (IS) 21
Chapter 4: Software and hardware Specifications 25
4.1 Software Specifications: 25
4.1 Hardware Specification 25
Chapter 5: Results 26
Chapter 6: Challenges Faced 29
Chapter 7: Future Enhancement 30
References 31
Bibliography and Datasets 33
Appendix 34

List of Figures

Figure 1L channel visualization..5
Figure 2ab channel visualization 5
Figure 3 Generative Adversarial Network...6
Figure 4 Conditional GAN..7
Figure 5 PatchGAN..12
Figure 6 VQVAE 15
Figure 7 Diffusion Process...15
Figure 8 Vision Transformer 16
Figure 9 Relative Positional Embedding process 16
Figure 10 Outpainting Architecture 18
Figure 11 Sampling Procedure...20
Figure 12 Result comparision of three different models 26
Figure 13 Outpainting results train and test sets..27

List of Tables

Table 2 Discriminator and Generator Loss function table 26
Table 1 Outpainting metrics 27

Acronyms

AI	:	Artificial Intelligence
ASPP	:	Atrous Spatial Pyramid Pooling
cGANs:	:	Conditional Generative Adversarial Networks
CNN	:	Convolutional Neural Networks
COCO	:	Common Objects in Context dataset
FID	:	Fréchet Inception Distance
GANs	:	Generative Adversarial Networks
IS	:	Inception Distance
PQDiff	:	Positional Query Diffusion
PSNR	:	Peak Signal to Noise Ratio
RGB	:	Red Green Blue
ReLU	:	Rectified Linear Unit
VQVAE	:	Vector Quantized Variational Autoencoder
ViT	:	Vision Transformer

Chapter 1: Introduction

1.1 Background

The rise of artificial intelligence (AI) technologies has significantly impacted various fields, including image processing and restoration. In the realm of cultural heritage preservation, AI offers exciting possibilities. This project explored the application of AI techniques for revitalizing Nepal's rich cultural heritage, as captured in historical photographs. We aimed to enhance appreciation for the nation's history and traditions by colourizing and potentially outpainting these archival materials.

Nepal possesses a remarkable tapestry of societal aspects, cultural traditions, and breathtaking landscapes, meticulously documented through photography for centuries. However, a significant amount of these invaluable records remains in black and white. This format can limit their visual impact and accessibility to modern audiences. By leveraging the power of AI, this project aimed to transform these monochromatic images into vibrant portrayals of Nepal's past, fostering a more profound understanding of the country's cultural evolution.

1.2 Objectives

The objectives of this project are:

- Implement AI techniques to colorize black and white historical photographs of Nepal
- Engagement with Nepalese culture and nature through image outpainting
- Ensure longevity and accessibility of Nepal's cultural heritage by leveraging AI-driven restoration methods.

1.3 Problem Statement

Historical Significance, Limited Accessibility: Historical photographs represent a critical element for record of Nepal's rich tapestry of society and culture. However, most of these images remain trapped in black and white, hindering their ability to resonate with and engage contemporary audiences. This limited accessibility stems from the lack of efficient colourization and expansion techniques specifically tailored to the unique context of Nepal's historical photographs.

Deterioration and Loss: The absence of proper restoration and enhancement methods jeopardize the well-being of their longevity. Without intervention, these photographs risk physical downturn and potential loss, depriving future generations of crucial insights into Nepal's cultural evolution.

Need for AI-driven Solutions: The project identified the need for AI-driven methods specifically designed to handle Nepal's historical photographs. By developing effective colourization and potential expansion algorithms, the project aimed to unlock the full potential of these archival materials. It would facilitate broader access, enhance preservation efforts, and promote the educational use of this vital historical record.

1.4 Motivation and Significance

Preserving and Celebrating Heritage: The project's motivation was the profound commitment to safeguarding and celebrating Nepal's vibrant cultural heritage for the upcoming generation. By leveraging AI-powered colourization and potential expansion techniques for historical photographs, we aimed to bridge the gap between past and present. It would allow contemporary audiences to engage more meaningfully with Nepal's diverse cultural tapestry.

Educational and Societal Value: The project recognized the inherent educational and societal value of visual representations of history. Colourized and potentially expanded photographs offer a significantly more accessible and engaging experience, unlike their black-and-white counterparts. This enhanced accessibility potentialized broaden historical understanding and foster a deeper appreciation for Nepal's past.

Transforming Perception and Identity: The project's significance lies in its potential to revolutionize how people perceive and interact with Nepal's cultural heritage. By harnessing the power of AI, we aimed to unlock hidden details and restore vibrancy to historical images. Furthermore, it not only deepens comprehension of Nepal's history, but also cultivates a more robust sense of cultural identity and pride among Nepali communities around the globe.

Interdisciplinary Collaboration: Furthermore, the project's integrative nature, combining computer science with cultural studies, underscores the importance of collaboration in tackling complex societal challenges. It demonstrates the value of advancing knowledge at the intersection of technology and the humanities.

Chapter 2: Related Works

The paper "Continuous-Multiple Image Outpainting in One Step via Positional Query and a Diffusion Based Approach" proposes PQDiff, a novel method for image outpainting. PQDiff addresses limitations in generating arbitrary expansions and avoids multi-step processes. It leverages positional information from random image crops during training and utilizes a diffusion-based generator with positional queries for inference. This approach outperforms existing methods on benchmark datasets, achieving superior FID scores and reducing computation time significantly.

Isola et al. (2017) explored conditional adversarial networks as a versatile approach for solving image-to-image translation tasks. Their framework not only learned to effectively map input images to desired output images but also autonomously acquired a loss function tailored for training this mapping. The release of their associated software, pix2pix, has further spurred widespread adoption and experimentation among numerous Twitter users, showcasing the system's impact on fostering artistic exploration in digital media.

Treneska et al. (2022) primarily focus on leveraging generative adversarial networks (GANs) for image colorization due to their capability in producing highly realistic colorized outputs. They propose employing conditional GANs (cGANs) specifically for this purpose and extend their findings to enhance performance in multilabel image classification and semantic segmentation tasks. Their empirical evaluations on the COCO and Pascal datasets reveal notable improvements, achieving a 5% increase in classification accuracy and a 2.5% enhancement in segmentation accuracy. These results underscore the effectiveness of image colorization with cGANs in enhancing downstream task performance without requiring additional manual annotations.

Chapter 3: Methodology

This section details the methodological approach for completion of the project. The project centred on leveraging the power of Generative Adversarial Networks (GANs) and Diffusion Models to revitalize historical photographs.

3.1 Image Colorization

Image colorization is the process of converting black and white images into their colorful states. It is still an area of research. A lot of research and models have been developed but fail to conserve their color integrity. Most papers use luminance–chrominance color spaces that allow us to separate the pixel intensity information from the pixel color information (Treneska et al., 2022). Here in this project, we follow the same color space and Pix-to-Pix architecture defined by Isola et al. (2017). They use conditional Generative Adversarial Networks to train their image colorization model where the generator part consists of UNET network, and the discriminator is PatchGAN.

3.2 Data Collection

We scrape the internet for our data collection. Sites like Pinterest, Twitter posts, and Facebook and Instagram posts were some of the sites we scraped data from. Our prime target of image colourization is colourizing old and vintage Nepalese photos for citation photos from the Rana regime, Prithvi Narayan Shah's great conquest, or frosty and foggy photos in people's storerooms and boxes. These photos are unorganized assets which were not available on the internet. However, some photographic sites played an instrumental role in finalizing our dataset. We have listed the sites in the ***Bibliography and Dataset section.*** Among them, ***archivenepal*** was the most conducive resource in this project.

In this project we created two different datasets and used a standard celeb_A dataset. One of the prepared datasets contained images that purely represent Nepalese contemporary society along with cultural, historical and political assets containing a total of 8666 samples of image. The second dataset was contaminated by samples of high-quality scenery dataset making it a total of 15,433 sample dataset. The celeb_A dataset contained 202,599 datasets, but we picked up 30,000 random images among them for training.

Also, we have 1,515 completely black and white images collected from different sources on which test predictions are performed.

3.3 Data Preprocessing

It is obvious that for any Deep Learning project, tremendous data preprocessing is required.

For colourizaton, we firstly convert **RGB** colourspace into CIELAB (LAB) colourspace. If the Images are not in **RGB** space, then first convert the images into **RGB** space and then into **LAB** space. In **LAB**, The **L** component stands for perceptual lightness with range [0, 100], meaning that it is the grayscale element. The **A** component represents the color position between red and green, while the **B** component represents the color position between blue and yellow; both components have ranges [−128, 127].

The intuition is that a luminance–chrominance color space is needed for the image colorization task to separate the intensity from the color information. The CIELAB (Lab) is one such color space used to describe all visible colors by the human eye. It was created to represent color changes in the same way as humans do. This means that a numeric change corresponds to a similar perceived difference in color. Space has little correlation between its three components. The **L** channel is used as an input to the model, while **A** and **B** channels are the target values.

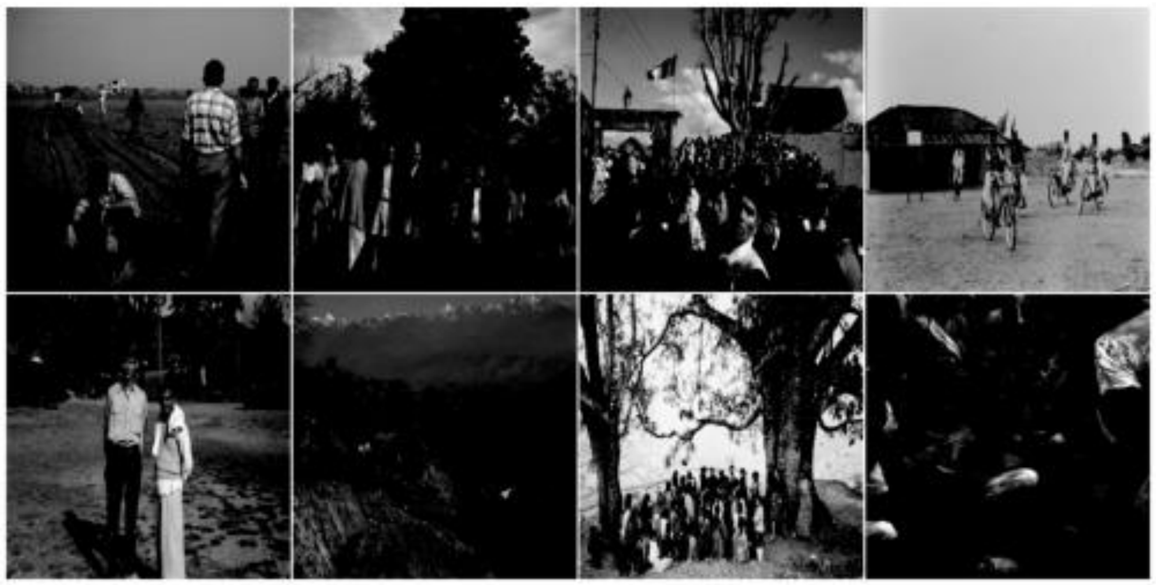


Figure 1L channel visualization

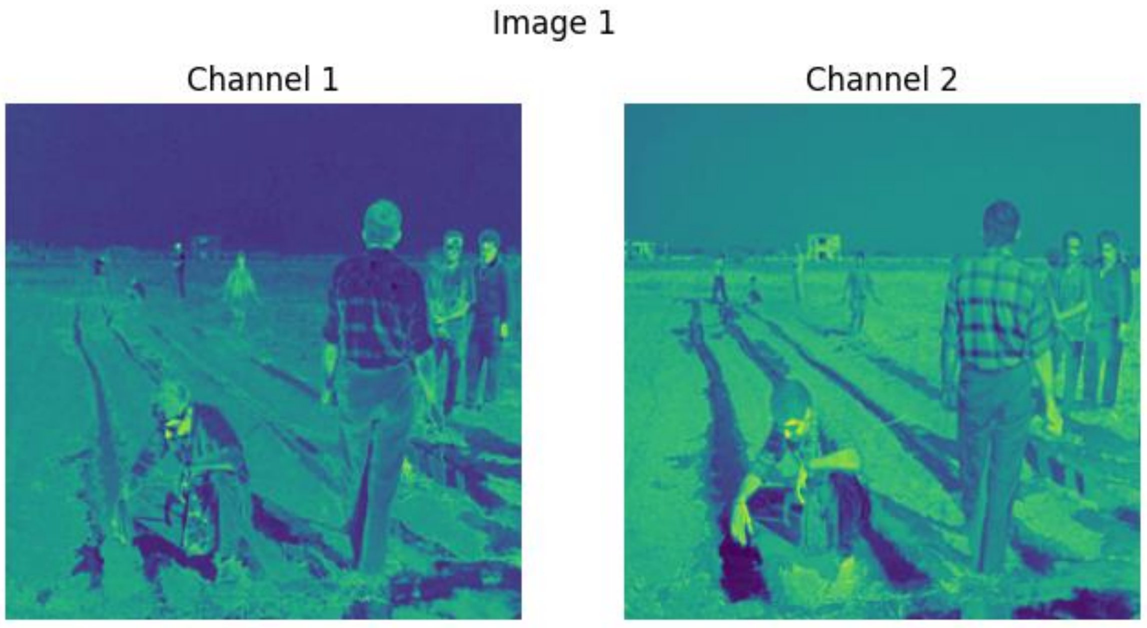


Figure 2ab channel visualization

3.4 Generative Adversarial Network

Generative Adversarial Networks (GANs) represent a class of generative models comprised of two neural networks with opposing objectives:

Generator: The generator network aims to produce realistic outputs that are indistinguishable from real-world data.

Discriminator: The discriminator network serves as a classifier, tasked with differentiating between genuine data and the generated outputs produced by the generator.

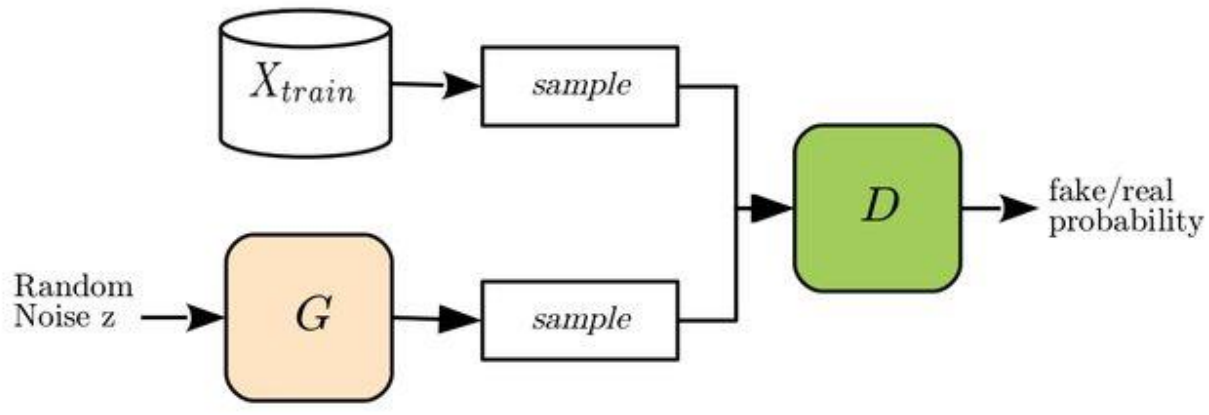


Figure 3 Generative Adversarial Network

Reference: https://www.researchgate.net/publication/317061929_LOGAN_Evaluating_Privacy_Leakage_of_Generative_Models_Using_Generative_Adversarial_Networks/figures?lo=1

The training process in a GAN is an iterative one. The generator continuously strives to improve its ability to create realistic outputs by attempting to deceive the discriminator. Simultaneously, the discriminator refines its ability to discern real data from generated outputs. This adversarial training process helps both networks achieve optimal performance:

The generator learns to produce increasingly realistic outputs that effectively mimic the underlying distribution of real data.

The discriminator becomes adept at identifying subtle differences between real data and generated outputs.

Conditional GANs condition the network on the black and white images and generate a color output image.

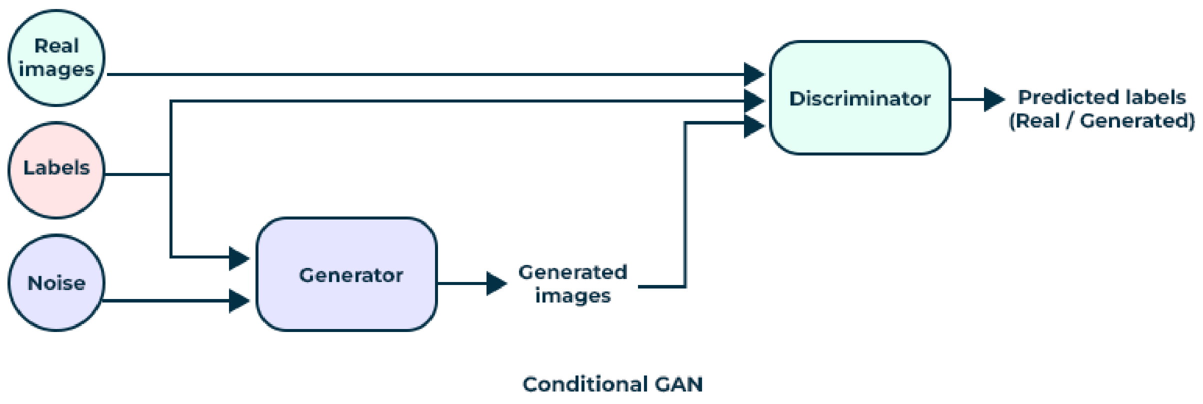


Figure 4 Conditional GAN

Reference: https://www.geeksforgeeks.org/conditional-generative-adversarial-network/

3.5 Objective Function

The objective functions used for training conditional generative adversarial networks is as follows:

$$\min \quad \max \quad V(G,D) = E_{x,y\sim p_{data}(x)}[\log D\,(x,y)] + E_{y,z\sim p_z(z)}\left[\log\left(1 - D\left(G(Z)\right)\right)\right]$$

The generator G tries to minimize the objective function while the discriminator D tries to maximize it, where x is the input grayscale image and y is the output for color channels.

3.5.1 Discriminator Backward Propagation

1. **Fake Image Discrimination**:
    - A fake image is generated by concatenating the grayscale input (L channel) and the colorized output produced by the generator.
    - The discriminator evaluates the fake image, producing fake_predictions.
    - The binary cross-entropy loss between fake_predictions and the target label False (indicating fake) is calculated using GANcriterion.
2. **Real Image Discrimination**:
    - A real image is created by concatenating the grayscale input (L channel) and the true color channels (ab channel).
    - The discriminator evaluates the real image, producing real_predictions.
    - The binary cross-entropy loss between real_predictions and the target label True (indicating real) is calculated using GANcriterion.

3. **Total Discriminator Loss**:
    - The total discriminator loss is the average of the fake loss and the real loss.

- The total loss is backpropagated to update the discriminator's weights.

3.5.2 Generator backpropagation

1. **Adversarial Loss**:
   - The generator aims to fool the discriminator into classifying its outputs as real.
   - A fake image is generated.
   - The discriminator evaluates the fake image, producing fake_predictions.
   - The binary cross-entropy loss between fake_prediction and the target label True (indicating real) is calculated
2. **L1 Loss**:
   - The L1 loss measures the absolute difference between the generator's output (fake_color) and the true color channels (ab).
   - This loss, scaled by lambda_L1, ensures that the colorization is accurate in terms of pixel values.
3. **Total Generator Loss**:
   - The total generator loss is the sum of the adversarial loss and the L1 loss
   - The total loss is back propagated to update the generator's weights.

3.6 UNET

In our first approach we followed Pix2Pix by Isola et al. (2017). We used UNET architecture for the generator and Patch Gan for the discriminator section. Isola et al. (2017) claim that the U-Net architecture has better results than that of an encoder-decoder network. A U-Net architecture allows low-level information to shortcut across the network i.e., UNET progressively downsamples the image, until a bottleneck, after which the process is reversed, and the image is upsampled to its original size. Skip connections are also added to facilitate the flow of low-level information through the network.

3.6.1 UNET Architecture Description

Down-sampling Path (Encoder):

- **Convolutional Layers**: The encoder consists of a series of convolutional layers, with each layer increasing the depth of the feature maps. Specifically, the architecture follows the sequence: c64→c128→c256→c512→c512→c512→c512→c512.

- **C64:** 64 filters, 4x4 kernel size, stride 2, padding 1
- **C128:** 128 filters, 4x4 kernel size, stride 2, padding 1

- **C256:** 256 filters, 4x4 kernel size, stride 2, padding 1

- **C512(5 Times):** 512 filters, 4x4 kernel size, stride 2, padding 1

Stride: The stride for all down-sampling layers is set to 2, which helps in progressively reducing the spatial dimensions of the input image while increasing the depth of the feature maps.

Leaky ReLU Activation: Leaky ReLU activations are used with a negative slope of 0.2. This choice allows a small gradient when the unit is not active, which helps mitigate the vanishing gradient problem often encountered in deep networks.

Batch Normalization: Batch normalization is applied after every convolutional layer except for the first one. This normalization technique stabilizes and accelerates training by normalizing the output of each layer to have a mean of zero and a standard deviation of one, helping to maintain a consistent scale of features across layers.

Bottleneck Layer:

Convolutional Layer: The bottleneck layer is the deepest point of the U-Net, consisting of one convolutional layer.

- **C512: 512 filters, 4x4 kernel size, stride 2, padding 1**

  Stride: The stride is set to 2, reducing the spatial dimensions to their minimum before up-sampling begins.

  Leaky ReLU Activation: The same Leaky ReLU activation with a negative slope of 0.2 is used, continuing to address the vanishing gradient problem.

  Batch Normalization: Batch normalization is applied, continuing to stabilize and accelerate training by maintaining a consistent scale of features.

Up-sampling Path (Decoder):

Convolutional Layers: The decoder mirrors the encoder but uses transposed convolutional layers for up-sampling. Specifically, the architecture follows the sequence: c512→c512→c512→c512→c256→c128→c64→cN.

- C512 (4 times): 512 filters, 4x4 kernel size, stride 2, padding 1
- C256: 256 filters, 4x4 kernel size, stride 2, padding 1
- C128: 128 filters, 4x4 kernel size, stride 2, padding 1
- C64: 64 filters, 4x4 kernel size, stride 2, padding 1
- CN (final layer): 2 filters, 4x4 kernel size, Stride: 2, Padding: 1

Stride: The stride for all up-sampling layers is set to 2, which helps in progressively increasing the spatial dimensions back to the original input size.

ReLU Activation: ReLU activations are used for upsampling layers, promoting positive activations and aiding in the reconstruction of the output image.

Batch Normalization: Batch normalization is applied after every transposed convolutional layer except for the final one. This normalization continues to stabilize and accelerate training by maintaining consistent feature scales.

Final Layer Activation: The final layer uses a Tanh activation function, which maps the output to a range between -1 and 1, suitable for generating images.

3.6.2 New Generator Architecture and Pretraining Strategy

We decided to pretrain the generator separately in a supervised and deterministic manner to avoid the problem of "the blind leading the blind" in the GAN game where neither generator nor discriminator knows anything about the task at the beginning of training.

Generator Architecture:

Backbone Selection:

- A pretrained ResNet18 model is chosen as the backbone for the generator's down-sampling path.
- ResNet18 is a widely used deep learning architecture originally trained on ImageNet for image classification tasks.

U-Net Construction:

- The U-Net architecture is extended from the ResNet18 backbone.
- This involves using the feature extraction capabilities of ResNet18 and adapting it to the needs of image colorization, where the goal is to predict color (ab channels) given the grayscale input (L channel).

Key Components:

- **Encoder (Down-sampling Path):** Utilizes layers from ResNet18 to progressively extract hierarchical features from input images.
- **Decoder (Up-sampling Path):** Mirrors the encoder but in reverse, reconstructing the full-resolution colorized image from extracted features.

3.6.3 Pretraining Strategy:

Stage 1: Backbone Pretraining for Classification

- The pretrained ResNet18 model is used as the backbone.

- It is initialized with weights learned from ImageNet, optimizing for high-level feature extraction.

Stage 2: Generator Pretraining for Colorization

- The entire generator (U-Net) is trained using a supervised approach with L1 loss.
- This step fine-tunes the ResNet18-based U-Net specifically for the task of colorization.
- The L1 loss function computes the mean absolute error between predicted colorized images and ground truth colorized images.

3.6.4 Integration into GAN Framework:

- After pretraining, the generator (U-Net) is saved with its weights (res18-unet.pt).
- In subsequent steps, the pretrained generator is integrated into a GAN setup to train on the actual target dataset using combined adversarial and L1 loss strategies.

3.7 PatchGAN (Markovian Discriminator)

PatchGAN motivates GAN to only model high-frequency structure. It tries to classify if each **Nunn** patch of image is true or fake. It is run convolutionally across all the images, averaging the values to return the final input.

3.7.1 Discriminator Architecture Description

- **Convolutional Layers**: The discriminator consists of a series of convolutional layers, with each layer increasing the depth of the feature maps. Specifically, the architecture follows the sequence: c64→c128→c256→c512.
    - c64: 64 filters, 4x4 kernel size, stride 2, padding 1
    - c128: 128 filters, 4x4 kernel size, stride 2, padding 1
    - c256: 256 filters, 4x4 kernel size, stride 2, padding 1
    - c512: 512 filters, 4x4 kernel size, stride 1, padding 1
- **Stride**: The stride for the final layer is set to 1, while all preceding layers use a stride of 2. This helps in reducing the spatial dimensions for the final decision, ensuring that the discriminator can make fine-grained decisions about the realism of local patches in the image.
- **Leaky ReLU Activation**: All ReLU activations in the network are replaced with Leaky ReLU activations with a negative slope of 0.2. This choice allows a small gradient when the unit is not active, which helps mitigate the vanishing gradient problem often encountered in deep networks.
- **Batch Normalization**: Batch normalization is applied after every convolutional layer except for the first and last one. This normalization technique stabilizes and accelerates training by normalizing the output of each layer to have a mean of zero and a standard deviation of one, which helps maintain a consistent scale of features across layers.

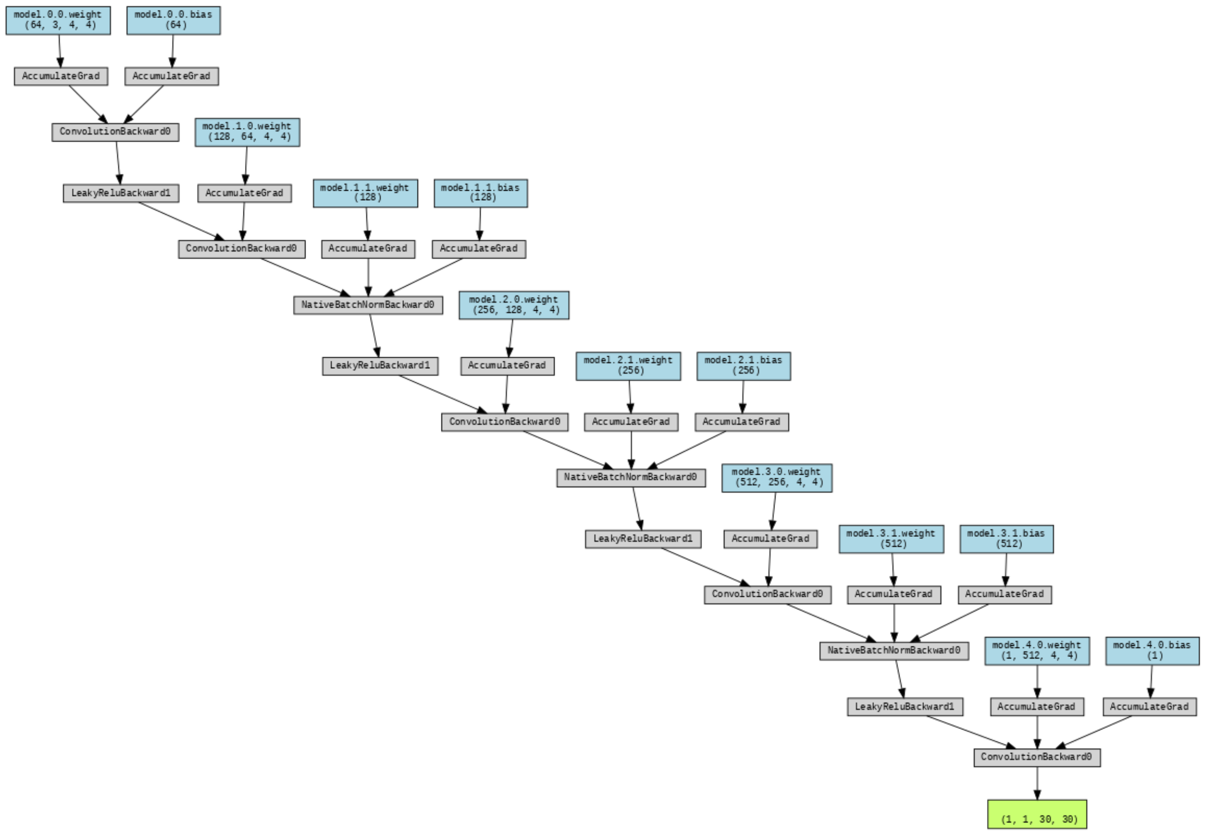


Figure 5 PatchGAN

- **Final Layer and Output:** After the last convolutional layer, the output is mapped to a one-dimensional feature using a final convolutional layer with no activation function (the activation function is embedded in the loss function). This layer outputs a score indicating whether the input patch is real or fake.

3.8 Deeplabv3 Generator

Originally developed for segmentation task, in this project we utilize this model to explore its generational power. The DeeLapv3 model uses Atrous Convolution Layers and Atrous Spatial Pyramid Pooling Layers to conserve local information of the image. Our hypothesis is that preserving local information of the image may lead to color conservation of original image. As, in UNET + PatchGAN model, the colourization results are promising, however fail to conserve the original colour.

3.8.1 Detailed DeepLabv3 Generator Architecture

1. **Encoder: ResNet-50**

   The encoder part of the model is based on the ResNet-50 architecture, which is known for its deep residual learning capabilities. Here are the key modifications and features:

- **Input Modification**: The first convolutional layer of the pre-trained ResNet-50 is modified to accept 1-channel input instead of the usual 3-channel RGB input.
- **Feature Extraction**: The layers of ResNet-50 up to the final layer before the fully connected layer are used. This extracts rich feature representations from the input images, resulting in an output feature map of size [batch size, 2048, H/32, W/32].

2. **Atrous Spatial Pyramid Pooling (ASPP)**

   The ASSP module is a crucial part of the DeepLabV3 architecture, designed to capture multi-scale information by applying convolutions with different dilation rates. The ASPP module in this implementation includes:

   - **1x1 Convolution**: A 1x1 convolution to reduce the input channels to a fixed number of output channels.
   - **3x3 Convolutions with Different Dilation Rates**: Three 3x3 convolutions with dilation rates of 3, 6, and 9 respectively, which helps in capturing multi-scale contextual information.
   - **Global Average Pooling**: A global average pooling layer followed by a 1x1 convolution and upsampling to match the spatial dimensions of the other feature maps.
   - **Concatenation**: All the feature maps are concatenated along the channel dimension.
   - **Final Convolution**: A final 1x1 convolution reduces the concatenated feature maps to the desired number of channels.

3. **Decoder**

   The decoder is designed to upsample the feature maps back to the original input resolution, facilitating pixel-wise classification. The decoder comprises:

   a. **Transposed Convolutional Layers**: A series of transposed convolutional layers are used for upsampling. Each layer is followed by batch normalization and ReLU activation.

   b. **Final Layer**: The final transposed convolutional layer maps the output to the desired number of channels, producing the segmentation map. In this implementation, the output is mapped to 2 channels, typically representing ab channels.

3.9 Image Outpainting

Image outpainting, also known as image extrapolation, is a fascinating area in the field of computer vision and artificial intelligence that involves generating parts of an image beyond its original boundaries. This technology has evolved significantly over the years, tracing its roots back to the broader concept of image inpainting, which focuses on reconstructing missing or damaged parts of images. Early research in the 1980s and 1990s laid the groundwork with basic algorithms that used pixel interpolation and texture synthesis techniques to fill gaps in images. The advent of deep learning in the 2010s marked a significant turning point, enabling more sophisticated and realistic outpainting through the use of convolutional neural networks (CNNs), generative adversarial networks (GANs) and Diffusion.

Key milestones in the development of image outpainting include the introduction of PatchMatch in 2009, which revolutionized image editing by allowing for efficient patch-based synthesis. This was followed by the application of GANs in image generation tasks, notably with the release of the Pix2Pix model in 2017, which highlighted the potential of conditional GANs for image-to-image translation. More recently, models like DALL-E, introduced by OpenAI in 2021, have demonstrated impressive capabilities in creating and extending images from textual descriptions, further pushing the boundaries of what is possible with image outpainting . These advancements have not only enhanced the technical capabilities of image outpainting but have also broadened its applications in fields such as art, design, and virtual reality, making it a valuable tool in creative and professional workflows

3.10 Data Preprocessing

To prepare our dataset for image colorization, we began by scraping various online sources, including Pinterest, Twitter, Facebook, Instagram, and specialized photographic sites such as **Archive Nepal**. We then meticulously checked for and removed any corrupt images to ensure data integrity. Finally, all images were converted into a consistent format, specifically JPG, in the dataset for subsequent processing and analysis.

3.11 Vector Quantized Variational Autoencoder

VQ-VAE is a type of variational autoencoder that uses vector quantization to obtain a discrete latent representation. It differs from VAEs in two keyways: the encoder network outputs discrete, rather than continuous, codes; and the prior is learnt rather than static. To learn discrete latent representation, ideas from vector quantization (VQ) are incorporated. Using the VQ method allows the model to circumvent issues of posterior collapse - where the latent are ignored when they are paired with a powerful autoregressive decoder - typically observed in the VAE framework. Pairing these representations with an autoregressive prior, the model can generate high quality images, videos, and speech as well as doing high quality speaker conversion and unsupervised learning of phonemes. The encoder consists of 2 strided convolutional layers with stride 2 and window size 4×4, followed by two residual 3×3 blocks (implemented as ReLU, 3x3 conv, ReLU, 1x1 conv), all having 256 hidden units. The decoder similarly has two residual 3×3 blocks, followed by two transposed convolutions with stride 2 and window size 4×4.

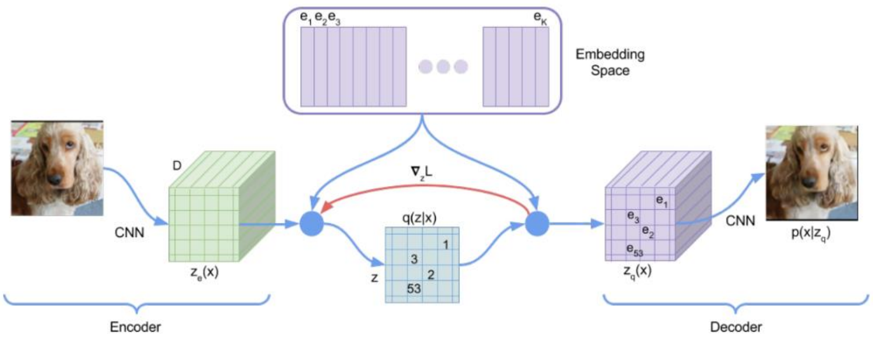


Figure 6 VQVAE

Reference: https://arxiv.org/pdf/1711.00937

3.12 Diffusion

Diffusion models are advanced machine learning approaches that uniquely generate high-quality data by progressively adding noise to a dataset and then learning to reverse this process. This innovative approach enables them to create remarkably accurate and detailed outputs, from lifelike images to coherent text sequences. Central to their function is the concept of gradually degrading data quality, only to reconstruct it to its original form or transform it into something new. This technique enhances the fidelity of generated data and offers new possibilities in areas like medical imaging, autonomous vehicles, and personalized AI assistants.

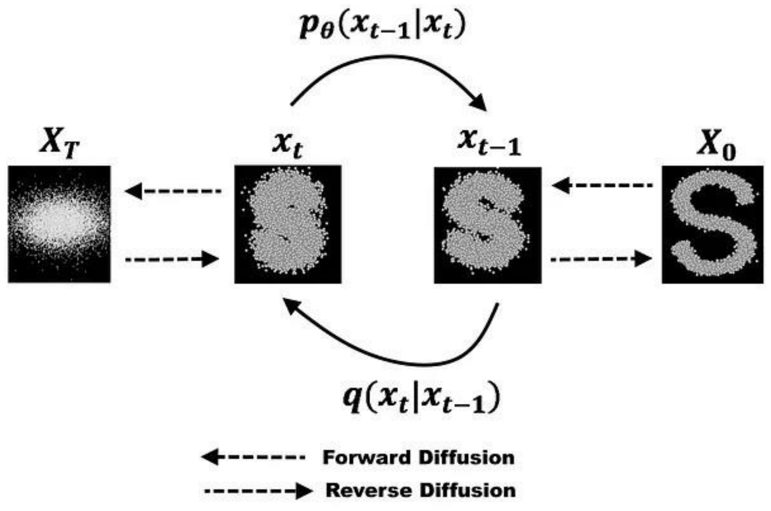
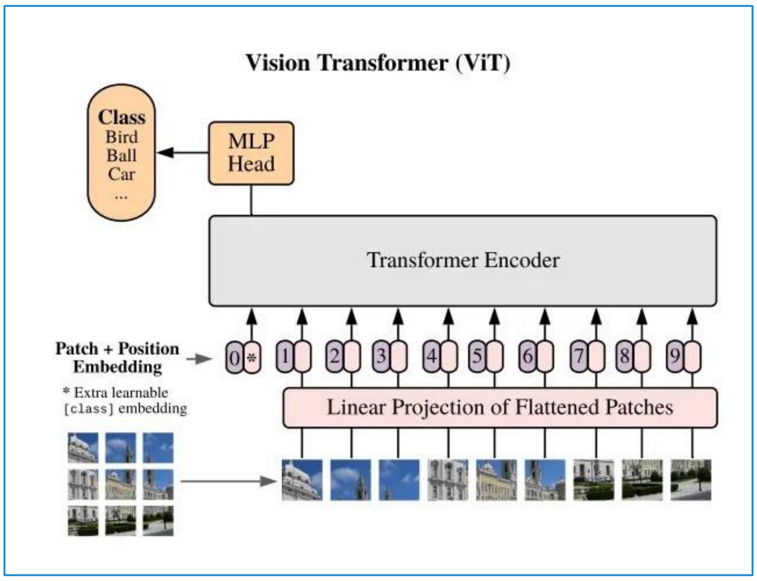


Figure 7 Diffusion Process

3.13 Vision Transformer

Vision Transformer (ViT) is a groundbreaking neural network architecture that reimagines how we process and understand images. The Vision Transformer (ViT) model was introduced in 2021 in a conference research paper titled "An Image is Worth 16*16 Words: Transformers for Image Recognition at Scale," published at ICLR 2021. Inspired by the success of Transformers in natural language processing, ViT introduces a new way to analyze images by dividing them into smaller patches and leveraging self-attention mechanisms. This allows the model to capture both local and global relationships within images, leading to impressive performance in various computer vision tasks.



Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches

Figure 8 Vision Transformer

Reference: https://medium.com/@hansahettiarachchi/unveiling-vision-transformers-revolutionizing-computer-vision-beyond-convolution-c410110ef061

3.14 Positional query diffusion model (PQDiff's)

Our approach for image outpainting is Positional query diffusion model (PQDiff's). This approach mainly consists of key modules: relative positional embedding, cross attention in the position-aware transformer model, and sampling pipeline.

3.14.1 Relative Positional Embedding

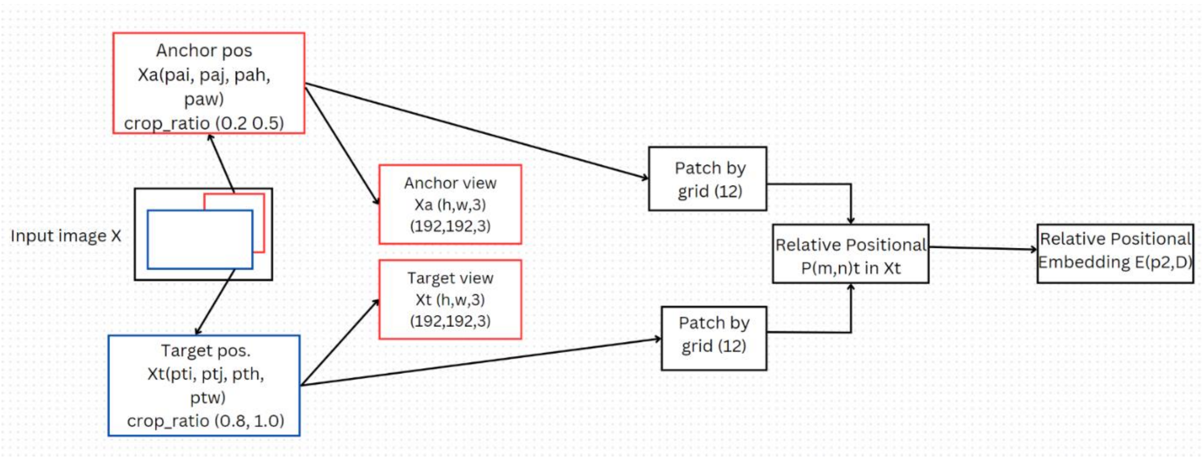


Figure 9 Relative Positional Embedding process

Given the image $X \in \mathbb{R}^{H\times W\times 3}$ from the training set, we first randomly crop the image twice and resize the cropped image to generate two views $X_\alpha \in \mathbb{R}^{h_1\times W_1\times 3}$ and$X_t \in \mathbb{R}^{h_1\times W_1\times 3}$, where X_α and X_t are denoted as the anchor and target view which are not necessarily the same size. We set the random crop ratio of the anchor view to (0.15, 0.5) and the ratio of the target view to (0.8, 1.0), aiming to use the small view to predict the larger view. Then we make the shape identical to both anchor and target view of shape (192, 192, and 3). According to random crop ratio we calculate the Positions$P_A = \{P_{Ai}, P_{Aj}, P_{Ah}, P_{Aw}\}$ (top location, left location, height and width), $P_T = \{P_{Ti}, P_{Tj}, P_{Th}, P_{Tw}\}$ of the two views X_A andX_T, and crop the training images. The objective is to calculate the relative position of each patch $\left[X_T^{(1)}, X_T^{(2)}, \ldots, X_T^{(K_T^2)}\right]$ in the target view based on the anchor viewX_A, where (K_T^2) is the number of patches in the target view. To calculate the number of patches we define the grid size where we set grid_size=12 and use anchor and target position height and width information. (Typically, we set$L = \frac{h\times w}{P^2}$) To specifically, we calculate the patch-level relative position of the target view via the following equation:

$$p_t^{m,n} = [p_t^m, p_t^n] = [\underbrace{\frac{K_a\cdot(p_{ti}-p_{ai})}{p_{ah}} + \frac{p_{th}\cdot K_a\cdot(m-1)}{K_t\cdot p_{ah}}}_{\text{Row}}, \underbrace{\frac{K_a\cdot(p_{tj}-p_{aj})}{p_{aw}} + \frac{p_{tw}\cdot K_a\cdot(n-1)}{K_t\cdot p_{aw}}}_{\text{Column}}]$$

Where K_α^2 means the number of patches of the anchor view. $p_t^{m,n}$Means the relative position of the patch located at mth row and nth column in the target view. Then, for each patch, we use the following popular form in transformers to generate the relative positional embedding:

$$\mathbf{P}_T^{m,n} = \left[\sin\left(\frac{p_t^m}{e^{2*1/d}}\right), \cos\left(\frac{p_t^m}{e^{2*2/d}}\right), \cdots, \sin\left(\frac{p_t^m}{e}\right),\right. \\ \left.\sin\left(\frac{p_t^n}{e^{2*1/d}}\right), \cos\left(\frac{p_t^n}{e^{2*2/d}}\right), \cdots, \sin\left(\frac{p_t^n}{e}\right)\right]$$

Also, we defined as:

$$\mathbf{E}_{m,n} = \left[\sin\left(\frac{m}{e^{2*1/d}}\right), \cos\left(\frac{m}{e^{2*2/d}}\right), \cdots, \sin\left(\frac{m}{e}\right), \sin\left(\frac{n}{e^{2*1/d}}\right), \cos\left(\frac{n}{e^{2*2/d}}\right), \cdots, \sin\left(\frac{n}{e}\right)\right],$$

Where e = 10,000 is the pre-defined parameter, (m,n) means the position of top-left path position. Note that we randomly crop two views, and as a result, the positional relationship of the two views might be either containing, overlapping, or non-overlapping, and our model can jointly learn these three relationships. Here we find the shape of our embedding is (p2, D) that is (144, 1024) where p2 is area of grid and D is self-defined Dimensionality of the token embedding's.

3.14.2 Forward and Backward of Diffusion

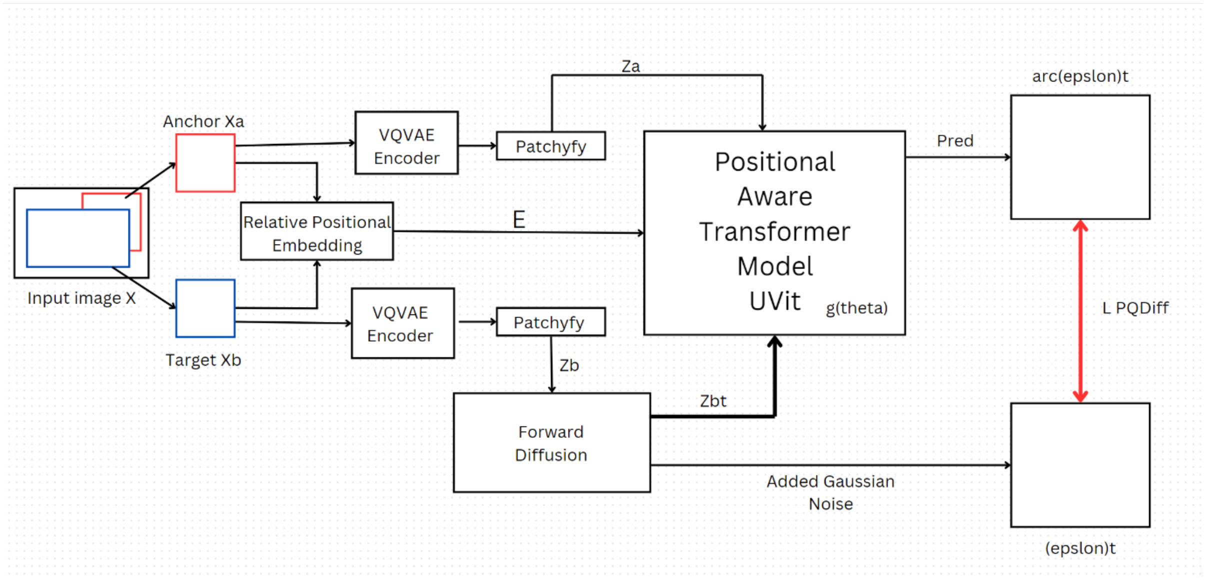


Figure 10 Outpainting Architecture

Given the anchor view $X_\alpha \in \mathbb{R}^{h\times w\times 3}$, target view $X_b \in \mathbb{R}^{h\times w\times 3}$ and the relative positional embedding $E \in \mathbb{R}^{L\times D}$ (L and D are the patches' area and the predefined dimension), we encode the two views by VQVAE (Van Den Oord et al., 2017) to compress the images into discrete latent space, resulting in $Z_\alpha \in \mathbb{R}^{h'\times w'\times c}$ and $Z_b \in \mathbb{R}^{h'\times w'\times c}$ (usually, h' < h, w' < w). The compression aims to improve the training efficiency and convergence speed of the diffusion models. After obtaining the two latent views, we patchify the two views and obtain the anchor sequence $\{Z_\alpha^1, Z_\alpha^2, \ldots, Z_\alpha^L\}$ and the target sequence $\{Z_b^1, Z_b^2, \ldots, Z_b^L\}$ (typically, we set$L = \frac{h\times w}{p^2}$). Then, the forward process of diffusion on the target sequence can be formulated as:

$$q(\mathbf{z}_{b_t}|\mathbf{z}_{b_{t-1}}) = \mathcal{N}(\mathbf{z}_{b_{t-1}}|\sqrt{\alpha_t}\mathbf{z}_{b_{t-1}}, \beta_t\mathbf{I}), \text{ and } \; q(\mathbf{z}_{b_t}|\mathbf{z}_{b_0}) = \mathcal{N}(\mathbf{z}_{b_t}|\sqrt{\overline{\alpha}_t}, (1-\overline{\alpha}_t)\mathbf{I}),$$

Where Z_{b0} are the original target sequences $\{Z_b^1, Z_b^2, \ldots, Z_b^L\}$ and$\overline{\alpha_t} = \Pi_{t=1}^{t}\alpha_i$. For the background process, we have a neural network g_θ the position aware transformer model, taking the noisy targetZ_{bt}, clean anchor sequence $Z_a$and relative positional embeddings E as input. Then, the network aims to predict the added noise ϵ_t on the target sequenceX_{bt}. Then, the objective of PQDiff can be written as:

$$\mathcal{L}_{PQDiff} = ||\widetilde{\epsilon}_t - \epsilon_t||_p^p, \quad and \quad \widetilde{\epsilon}_t = g_\theta(\mathbf{x}_a, \mathbf{x}_{b_t}, \mathbf{E}, t).$$

We set p = 2 in line with the previous generative methods.

3.15 The Positional-Aware Transformer Model g_θ

Here, we describe the architecture of the neural network used in the diffusion model in detail. We have the noise targetZ_{bt}, clean anchorZ_α, relative positional embeddings E and the timestep t, we first concatenate the noisy target and the anchor sequence at the channel dimension, the concatenate of the noisy target and the clean anchor during backward diffusion to integrate information from both signals. This combined input helps the model learn and predict the characteristics of noise more effectively by comparing the noisy target with a reference (clean anchor), followed by a linear layer to map to original dimension, the aim of passing this concatenated vector through a linear layer is to reduce its dimensionality. This means transforming the concatenated vector from its original high-dimensional space to a lower-dimensional space. And we denote the mapped embedding as$Z \in L \times D$, Where D is predefined hidden dimension in the transformer network. Then, we feed the Z_g into the transformer encoder, which is composed of several transformer blocks. After the transformer encoder, the position-aware cross-attention mechanism is proposed to learn positional relationship between elements in Z_{bt} and. This helps in understanding how different parts of the sequences relate spatially or contextually, which is crucial for tasks like image generation. It can be formulated as:

$$\mathbf{z}_d = \mathrm{Attn}\left(\mathbf{Q}_\mathbf{E}, \mathbf{K}_{\mathbf{z}_g}, \mathbf{V}_{\mathbf{z}_g}\right) = \mathrm{Softmax}\left(\frac{\mathbf{Q}_\mathbf{E}\mathbf{K}_{\mathbf{z_g}}^\top}{\sqrt{D}}\right)\mathbf{V}_{\mathbf{z}_g},$$

Where

$\mathbf{Q}_E = \mathbf{E}\mathbf{W}_q$, $\mathbf{K}_{\mathbf{z}_g} = \mathbf{z}_g\mathbf{W}_k$, $\mathbf{V}_{\mathbf{z}_g} = \mathbf{z}_g\mathbf{W}_v$, and $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$ are learnable parameters, after capturing the information of the target positionZ_d , we directly feed the Z_d into the transformer decoder composed of several transformer blocks, followed by a convolutional layer to predict noise.

3.16 Sampling Pipeline

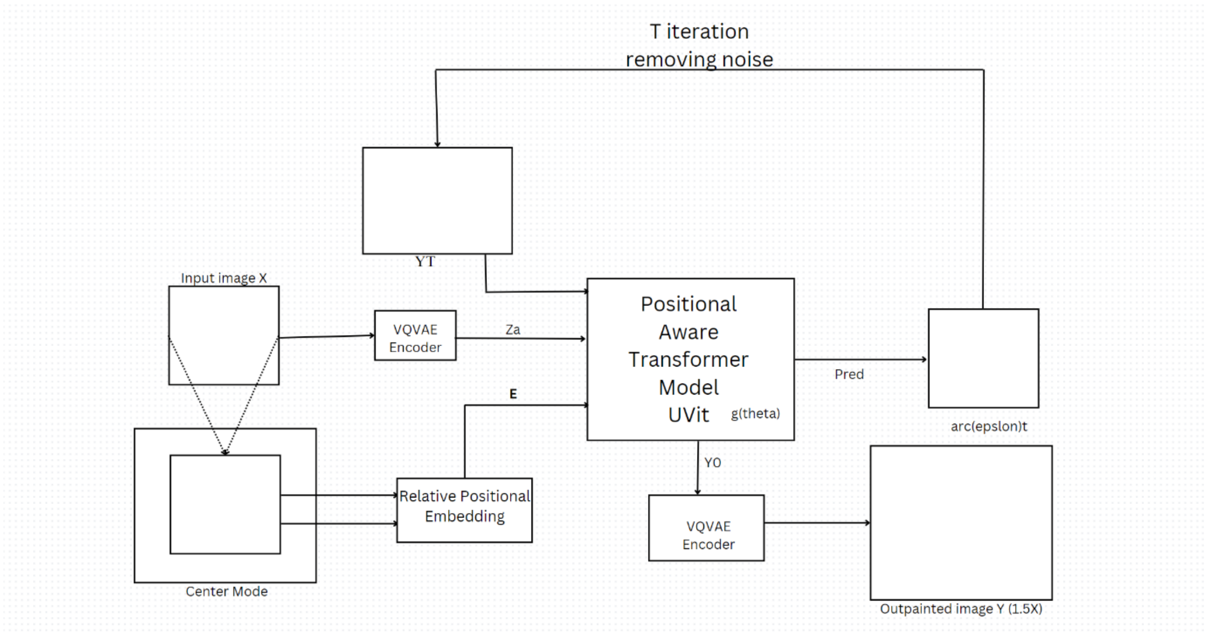


Figure 11 Sampling Procedure

After training the network well, we can outpaint the image in any controlled multiples, since the designed relative positional encoding can represent any positional relationship between two images. In the sampling stage, we can simply take the input sub-image as the anchor view and input any position we want. Then, we calculate the positional encoding of the given position and feed the RPE to the network. Then, the network can predict the noise as mentioned. We can simply compute the fake$\overline{Z}_{b0}$, and predict Z_{bt-1} step-by-step by:

$$q(\mathbf{z}_{b_{t-1}}|\mathbf{z}_{b_t},\widetilde{\mathbf{z}}_{b_0}) = \mathcal{N}(\mathbf{z}_{b_{t-1}};\widetilde{\mu}_t(\mathbf{z}_{b_t},\widetilde{\mathbf{z}}_{b_0}),\widetilde{\beta}_t\mathbf{I}),$$

$$\widetilde{\mu}_t(\mathbf{z}_{b_t},\widetilde{\mathbf{z}}_{b_0}) = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}\widetilde{\mathbf{z}}_{b_0} + \frac{\sqrt{\alpha}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{z}_{\mathbf{b_t}},\ and\ \ \widetilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t.$$

After several iterations, when t = 0, we can obtain the extrapolated images.

3.17 Training Details

We implement our approach with PyTorch on a platform equipped with RTX 3080 8 GB VRAM GPUs. The encoder is composed of 8 stacked transformer blocks. Then, a cross-attention block is employed, followed by the decoder made of 4 transformer blocks. Finally, a 3x3 convolutional layer is adopted to smooth the generated image. We adopt AdamW optimizer (Loshchilov & Hutter, 2019), and we set the learning rate to 0.0002, weight decay to 0.03, and betas to 0.99. In the training stage, we set the random crop ratio of the anchor view to (0.2, 0.5) and the ratio of the target view to (0.8, 1.0), aiming to use the small view to predict the larger view. We train PQDiff 300k iterations with 4 images per GPU on the Scenery and our Data. We set the resolution of each cropped view as 192x192. Our core idea of PQ-Diff is to utilize the randomly cropped two views (anchor and target) to learn the positional relation between them.

3.18 Evaluation and Baselines

We use the Inception Score (IS) (Salimans et al., 2016), Fréchet Inception Distance (FID) (Heusel et al, 2017) to measure the generative quality.

3.18.1 Fréchet Inception Distance

The FID score measures the distance between the feature vectors of real and generated images. These feature vectors are extracted from a pre-trained Inception v3 network.

A lower FID score indicates the generated images are more like the real images, meaning the generative model produces higher quality, more realistic outputs.

The FID compares the mean and standard deviation of the deepest layer in Inception v3. These layers are closer to output nodes that correspond to real-world objects such as a specific breed of dog or an airplane, and further from the shallow layers near the input image.

The FID for images are defined as

$$d_F(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma'))^2 = \|\mu - \mu'\|_2^2 + \operatorname{tr}\left(\Sigma + \Sigma' - 2(\Sigma\Sigma')^{\frac{1}{2}}\right)$$

in which the μ is the average value of multidimensional distribution. In the Image topic, it is the distribution of the activations of the last layer in Inception V3 model from a set of real-world images. Correspondingly, the μ' is the activations from a set of generated images. For example, it could be the images from the Stable Diffusion. The FID score we obtain after evaluating 500 images as test data is 90.3695

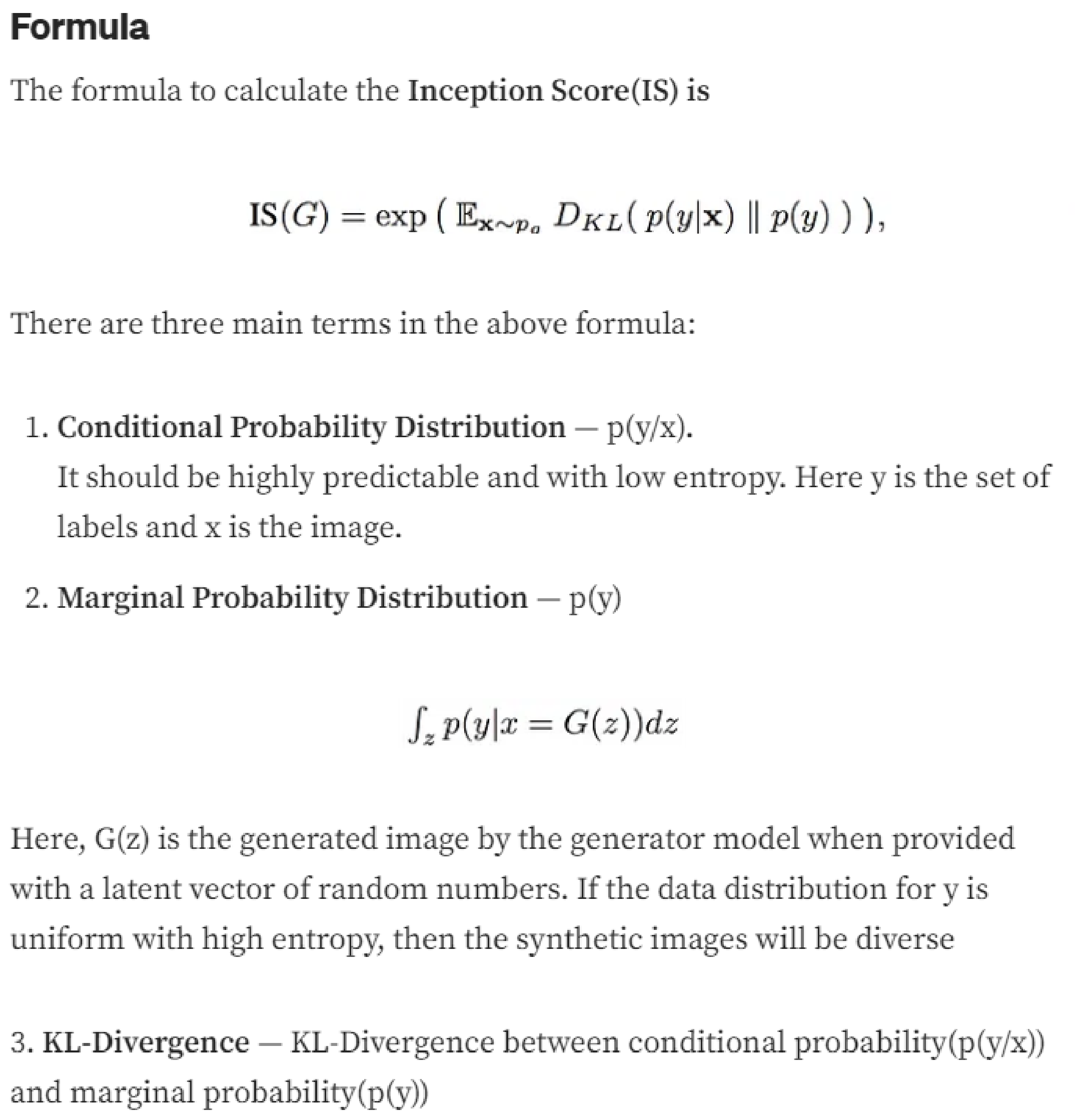
3.18.2 Inception Score (IS)

The Inception Score (IS) is an objective performance metric, used to evaluate the quality of generated images or synthetic images, generated by Generative Models. It measures how realistic and diverse the output images are. It can be used instead of subjective evaluation by humans. After FID (Frechlet Inception Distance), it is the second most important evaluation performance metric.

It measures two things:

1. **Diversity (Variety)** — How diverse the generated images are —the entropy of the overall distribution should be high.
2. **Quality (Goodness)** — how good the generated images are — Low entropy with high predictability is required.

The lowest IS can be zero and the highest IS can be infinite. Higher IS is always better.



Formula

The formula to calculate the **Inception Score(IS) is**

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(\, p(y|\mathbf{x}) \parallel p(y) \,) \right),$$

There are three main terms in the above formula:

1. **Conditional Probability Distribution** — p(y/x).
   It should be highly predictable and with low entropy. Here y is the set of labels and x is the image.
2. **Marginal Probability Distribution** — p(y)

$$\int_z p(y|x = G(z))dz$$

Here, G(z) is the generated image by the generator model when provided with a latent vector of random numbers. If the data distribution for y is uniform with high entropy, then the synthetic images will be diverse

3. **KL-Divergence** — KL-Divergence between conditional probability(p(y/x)) and marginal probability(p(y))

The inception Score we get is 3.0521

Peak Signal-to-Noise the Peak Signal-to-Noise Ratio (PSNR) is a widely used metric to assess the quality of reconstructed images, particularly in the fields of image compression and denoising. It measures the ratio between the maximum possible power of a signal (image) and the power of corrupting noise that affects the quality of its representation.

Here's a detailed look at PSNR: Ratio (PSNR):

Calculation of PSNR

1. **Mean Squared Error (MSE):**

   - First, compute the Mean Squared Error (MSE) between the original image I and the reconstructed (or generated) image K. For an image of size $m \times n$, MSE is calculated as:

$$\mathrm{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

2. **PSNR Calculation:**

   - The PSNR is then computed using the following formula:

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{\mathrm{MSE}} \right)$$

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

Interpretation of PSNR

- **Higher is Better:**
  - A higher PSNR value indicates better quality of the reconstructed image, as it means the noise level (error) is lower.
  - Commonly, PSNR values are measured in decibels (dB).

The PSNR score value we obtain is 21.6127 dB

Based on the evaluation of our diffusion model on 500 test images, the results indicate moderate performance across various quality metrics. A Fréchet Inception Distance (FID) score of 90.3695 suggests a noticeable difference between the distributions of the generated images and the real images, indicating room for improvement in the generative quality and diversity of our model. The Inception Score (IS) of 3.0521 reflects moderate performance, with the generated images having some diversity but not reaching the quality levels seen in more advanced generative models. These scores suggest that while our model can produce varied images, the similarity to real images needs enhancement.

The Peak Signal-to-Noise Ratio (PSNR) score of 21.6127, while useful in contexts of image reconstruction and compression, also highlights that the generated images have significant noise and visual artifacts compared to the original images. This score falls within a range that indicates considerable perceptual differences are likely noticeable. Therefore, while our model demonstrates the fundamental ability to generate images, the relatively high FID and low IS and PSNR scores suggest focusing on refining the model architecture and training process to enhance both the fidelity and diversity of the generated outputs.

Chapter 4: Software and hardware Specifications

For the successful implementation of this project, robust hardware and software systems are imperative due to the demanding nature of complex matrix operations on higher-dimensional data.

So, here are the software and hardware consumed till the completion of this project.

4.1 Software Specifications:

IDE (Vscode)

Python: 3.9, 3.10

Pytorch: 2.2.x

PIL,

Opencv-python,

matplotlib,

numpy,

google colab

4.1 Hardware Specification

RTX 3080 Ti 8GB,

RTX 1660 Ti 6GB

RAM: 16GB

Chapter 5: Results

In this study, we explored the application of conditional generative adversarial networks (cGANs) for image colorization. Our experiment was conducted on self-prepared datasets and a standard celebA dataset.

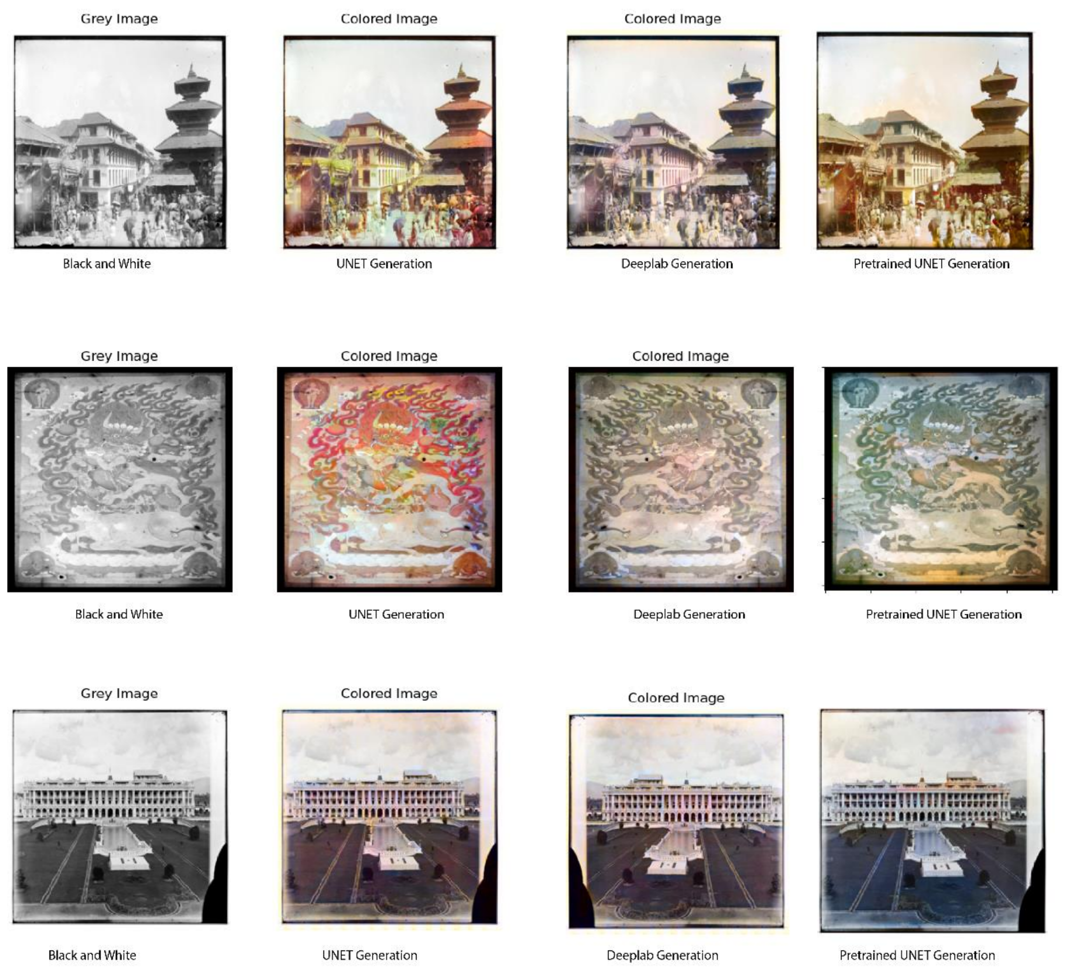


Figure 12 Result comparision of three different models

Table 1 Discriminator and Generator Loss function table

	Unet Scratch	**Deeplabv3**	**Unet Pretrained**
Discriminator loss	0.3402	0.003	0.61921
Generator Loss	6.5659	24.537	4.42224[i]

In the above results figure, what we can conclude is the UNET + PatchGAN works better than Deeplabv3 model. Also, in the rightmost image, we can see the best results using pretrained UNET generator with resent18 backbone.

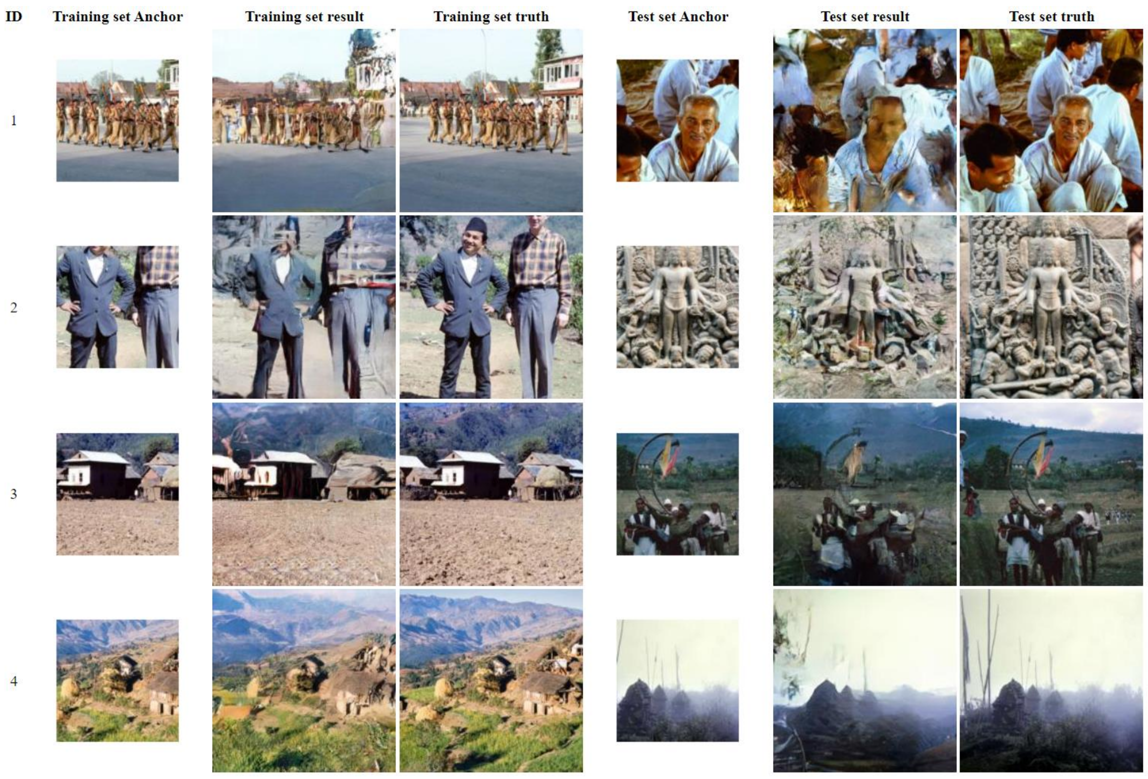


Figure 13 Outpainting results train and test sets

The results of the diffusion model, as seen in the provided images, demonstrate a generally good reproduction of the training and test set anchors. Through visual perception, we can say that the model captures the overall structure and color schemes effectively. However, several challenges are notable. Generated images often suffer from blurriness and a lack of finer details compared to the ground truth, particularly evident in intricate scenes like stone carvings and group portraits. Additionally, some artifacts are present, affecting the overall realism. Despite these issues, the model's ability to generate visually coherent images indicates strong potential, with room for improvement in enhancing detail and reducing artifacts for better fidelity.

However, the below metrics table suggests that the model consistently performs poorly.

Table 2 Outpainting metrics

Image	**FID**	**IS**	**PSNR**[ii]
1.5x	90.3695	3.0521	21.6127

Here the FID score obtained is **90.3695**, suggesting a noticeable difference between the generated and real images, indicating room for improvement in the model's generative quality. Also, the IS score here is **3.0521**, reflecting moderate performance with some diversity in the generated images but not reaching the quality levels of more advanced models. The PSNR value reported is **21.6127 dB**, which suggests that the generated images have significant noise and visual artifacts compared to the original images.

Chapter 6: Challenges Faced

Image to Image generation was one of our first try projects, and we enjoyed this project's journey. However, we faced some challenges that played a crucial role in hindering our project. Because of these challenges and complications, we could not make this project to its fullest potential.

1. Computational Resource Constraint
    - The training process for GANs and diffusion models is computationally intensive, necessitating the use of high-performance GPUs and significant memory resources. Managing computational resources effectively to accommodate large datasets and complex models was a logistical challenge, often leading to extended training times and the need for efficient resource allocation strategies. Besides, limited batch size led to poor quality results as we can see in the paper.
2. Niche Data quality
    - Ensuring the quality and consistency of the input images was a significant challenge. The datasets required extensive preprocessing to handle variations in resolution, lighting, and noise. Poor quality images could lead to suboptimal training of the models, impacting the overall performance of the model.
3. Training Stability:
    - Training GANs, particularly conditional GANs and diffusion models, posed challenges due to their inherent instability.
4. Steep Learning Curve:
    - Diffusion based models and Vision Transformer have steep learning curves. Hence management of academic and the semester project learning curve was challenging.

Chapter 7: Future Enhancement

Looking ahead, several avenues for future enhancements could significantly elevate the capabilities and performance of our current model. Here are some of the steps to be taken to enhance the project in future:

1. Use Denser UNET layers
2. The decoder part of the Deeplabv3 is coarse. So, working on the decoder part of DeepLabv3 may improve the generalizability of the model.
3. Using ImageGAN instead of PATCHGAN
4. Video Colourization
5. Using Autoencoder and Variational Autoencoder for colourization
6. Using multimodal approach for combining both colourizing and outpainting tasks
7. Using larger volume of data for better outpainting results

References

Zhang, S., Huang, J., Zhou, Q., Wang, Z., Wang, F., Luo, J., & Yan, J. (2024). CONTINUOUS-MULTIPLE IMAGE OUTPAINTING IN ONESTEP VIA POSITIONAL QUERY AND a DIFFUSIONBASED APPROACH. *ICLR*. https://arxiv.org/pdf/2401.15652

Treneska, S., Zdravevski, E., Pires, I. M., Lameski, P., & Gievska, S. (2022). GAN-Based Image colorization for Self-Supervised Visual Feature Learning. *Sensors*, *22*(4), 1599. https://doi.org/10.3390/s22041599

Isola, P., Zhu, J., Zhou, T., & Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* https://doi.org/10.1109/cvpr.2017.632

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 834–848. https://doi.org/10.1109/tpami.2017.2699184

Oord, A. V. D., Vinyals, O., & Kavukcuoglu, K. (2018). Neural Discrete Representation Learning. *Arxiv*. https://arxiv.org/pdf/1711.00937

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *Arxiv*. https://openreview.net/pdf?id=YicbFdNTTy

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-Attention with relative position representations. *Arxiv*. https://doi.org/10.18653/v1/n18-2074

Loshchilov, I., & Hutter, F. (2019). DECOUPLED WEIGHT DECAY REGULARIZATION. *csLG*. https://arxiv.org/pdf/1711.05101

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. *csLG*. https://arxiv.org/pdf/1606.03498

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs trained by a two Time-Scale update rule converge to a local Nash equilibrium. *arXiv (Cornell University)*, *30*, 6626–6637. https://arxiv.org/pdf/1706.08500

Bibliography and Datasets

CelebA Datasets:
https://www.kaggle.com/datasets/zuozhaorui/celeba?select=img_align_celeba

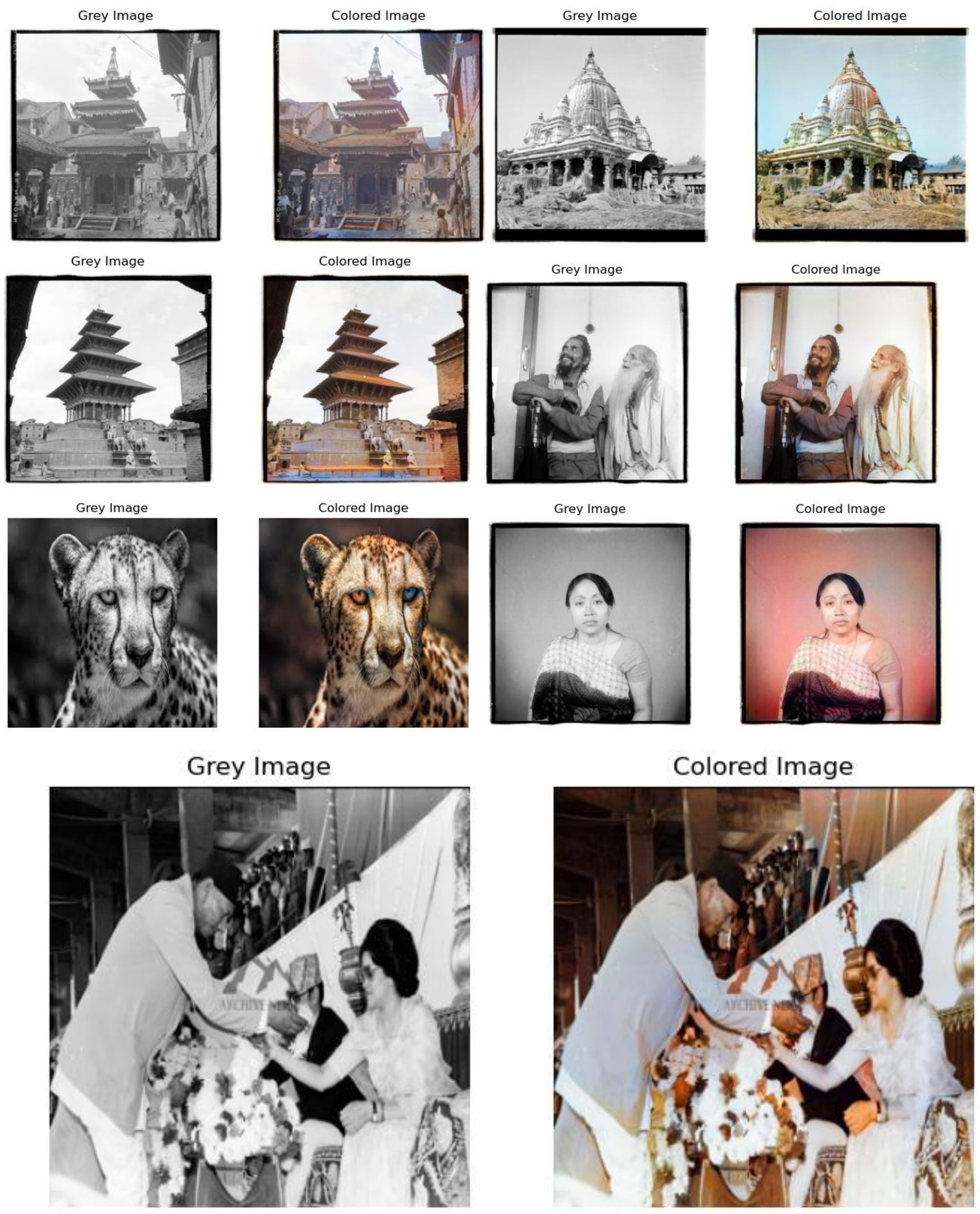
Scenery Datasets: https://www.kaggle.com/datasets/arnaud58/landscape-pictures

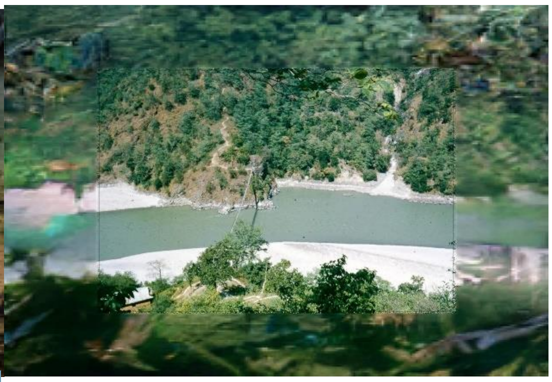
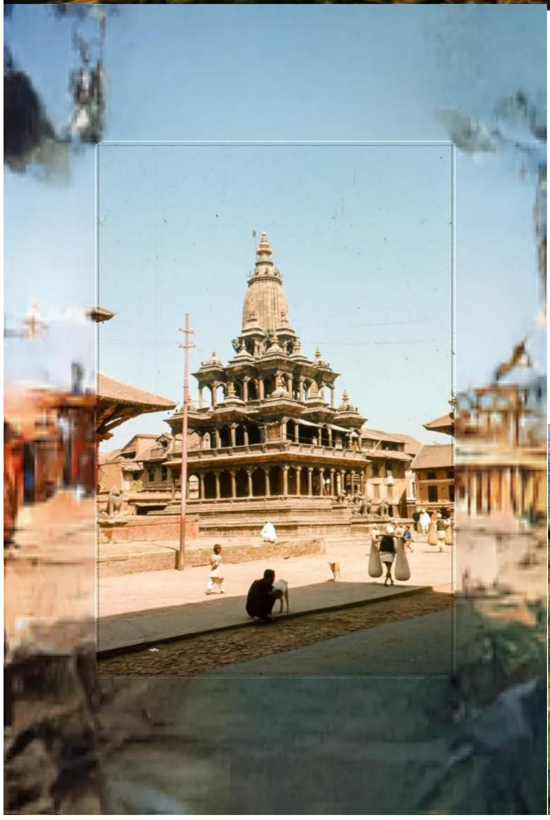
Scrapes Sites:

1. https://www.pinterest.com/subenjabegu/vintage-nepal/
2. https://nepal8thwonder.com/old-photos-of-nepal
3. https://www.gettyimages.com/photos/nepal-old-men
4. https://depositphotos.com/photos/old-nepal.html
5. https://www.alamy.com/stock-photo/vintage-nepal.html?sortBy=relevant
6. https://www.istockphoto.com/photos/panoramic-view-of-kathmandu-nepal
7. https://nepalitimes.com/here-now/making-nepal-s-history-colourful
8. https://depositphotos.com/photos/nepalese-old-village.html

Archive Nepal: https://www.archivenepal.org/home

Appendix


Grey Image
Colored Image
Grey Image
Colored Image
Grey Image
Colored Image
Grey Image
Colored Image
Grey Image
Colored Image
Grey Image
Colored Image
Grey Image
Colored Image

[i] Discriminator and Generator loss function table
[ii] outpainting metrices