Image Colorization using GAN's

Pranav Jain -210101078
Sreehari C -210101101

Introduction

- Recent advancements in deep learning have revolutionized the process of colorizing black and white images.
- Traditional methods required significant human input and hardcoding, but AI now enables an end-to-end solution.
- Despite assumptions of needing large datasets and lengthy training times, a new approach has been developed.
- Key aspects covered include model architectures, loss functions, training strategies, and code implementation.
- We have used PyTorch library to make and test the model

Motivation

- AI-Driven Colorization: The project aims to showcase how deep learning, specifically AI, can automate the process of colorizing black and white images, which was previously a manual and labor-intensive task.

- Efficient Strategy: It presents an efficient strategy for training a colorization model with the latest advances in deep learning, using a small dataset and short training times.

- Practical Application: The code serves as a practical application of image-to-image translation techniques in deep learning, particularly using Adversarial Networks (pix2pix) for colorization tasks.

Application/Use case

- Restoration of Historical Photographs.
- 2. digitally restore or enhance black and white films.
- Colorizing medical images such as X-rays or MRI scans.
- Can enhance the realism of virtual environments in VR simulations and games.
- Colorizing forensic images such as surveillance footage or crime scene.

Brief Literature

- Richard Zhang, Phillip Isola, Alexei A. Efros : Colorful Image Colorization (2015)
- Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 415–423
- Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. (2014)

Objectives

The main objective of this project was to explore how GAN’s can be used for image colorization. We implemented the pix2pix paper mentioned above using pytorch and tried to re-create the results.

GAN’s: A GAN model comprises of 2 sub-models:

- The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator.
- The discriminator learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

Methodology

We simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game

Generator: Our Generator consists of **U-net** layer. Firstly, we convert the image to **L,a,b** format and give the L part as input and the output will be again an **L,a,b** image.

Discriminator: Implements a model by stacking blocks of **Conv-BatchNorm-LeackyReLU** to decide whether the input image is fake or real at the end (classification problem)

Training: We simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game.

Loss Function:

- **Loss function we optimize** :

$$G^* = argmin_G max_D L_{cGAN}(G, D) + \lambda L_{L1}(G) \tag{1}$$

- **L1 Loss** :

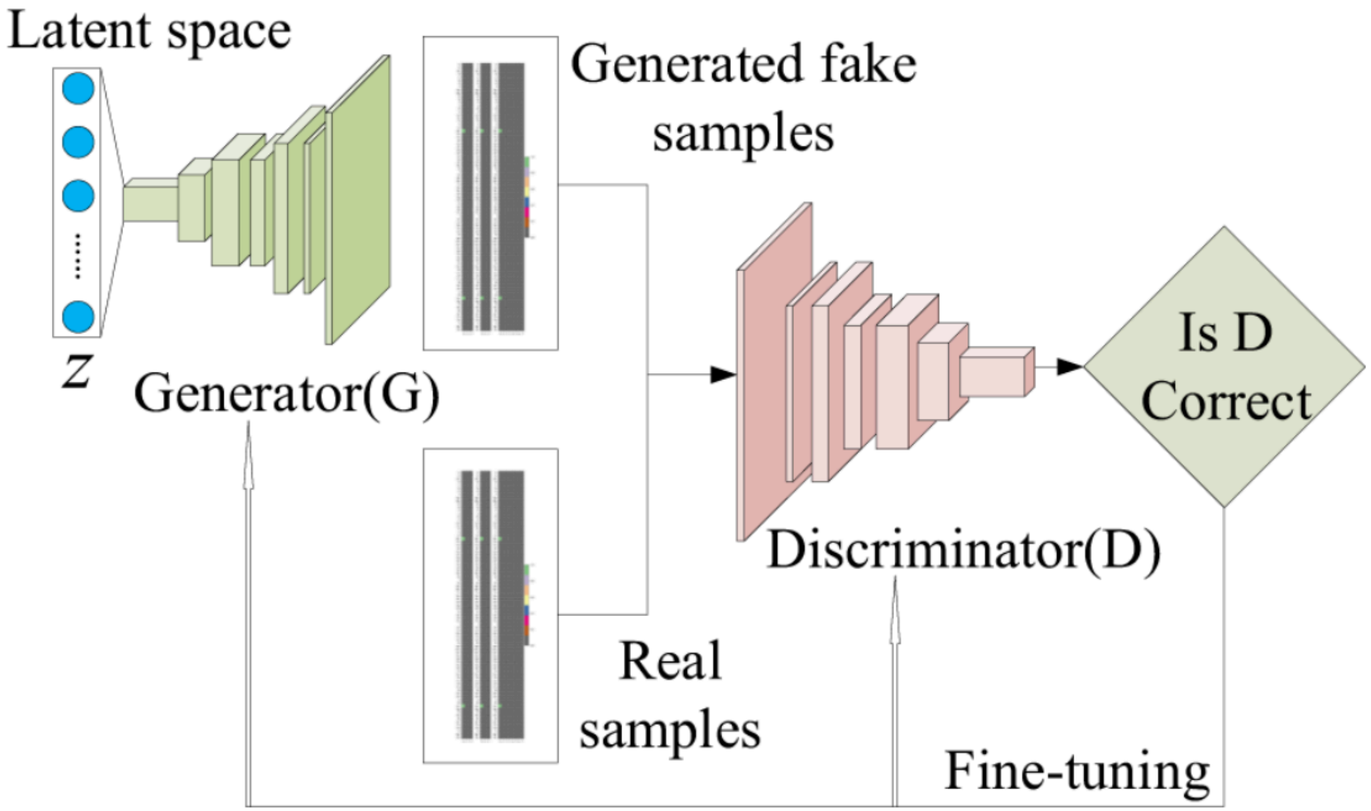
$$L_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1] \tag{2}$$

- **GAN Loss** :

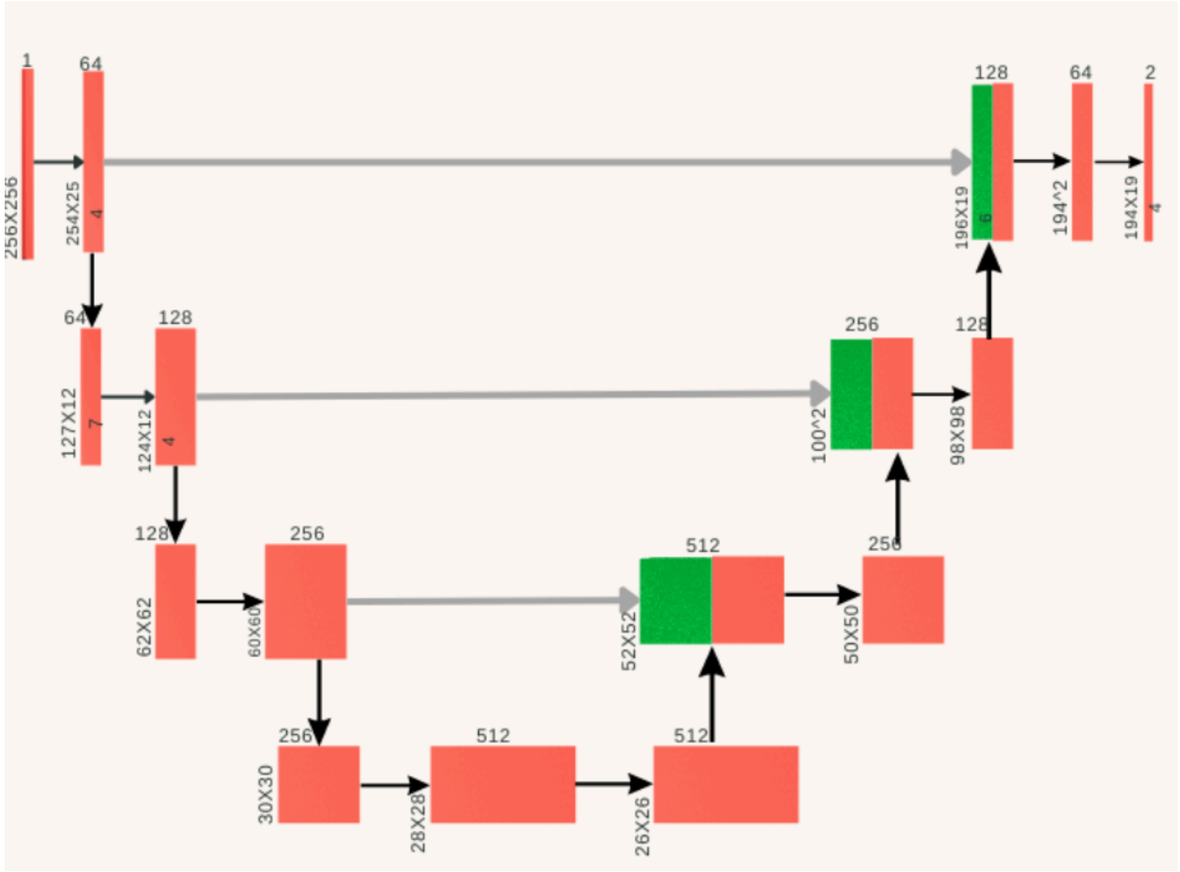
$$L_{cGAN}(G, D) = \mathbb{E}_{x,y}[logD(x, y)] + \mathbb{E}_{x,z}[log(1 - D(x, G(x, z))] \tag{3}$$

Where **x** as the grayscale image, **z** as the input noise for the generator, and **y** as the 2-channel output we want from the generator (it can also represent the 2 color channels of a real image). Also, **G** is the generator model and **D** is the discriminator.

Model Architecture:



Architecture of the Unet used in Generator:



Data

- Source for the dataset and train/val splits - We'll be using a subset of COCO dataset: 8000 training images and 2000 validation images.
- Preprocessing - Convert **RGB** image to **Lab** format as generator requires black and white image, obtained from L part.
- Sample data points/images/sound samples/text:




Results

Summary

- Image Colorization with AI: The research focuses on colorizing black and white images using U-Net and GAN (Generative Adversarial Networks) architectures.

- Efficient Training Strategy: The study presents an efficient training strategy that requires a smaller dataset and shorter training times, leveraging the latest advances in deep learning.

- GAN Implementation: The tutorial includes a detailed implementation of the U-Net model and GAN framework, explaining the colorization problem and the mathematical concepts behind the models.

- Pretraining and Results: The research highlights the importance of pretraining the generator model for improved colorization results and demonstrates significant advancements over traditional methods.