# Image Colorization using cGAN

Semester III Project

# Contents

# Introduction

# INTRODUCTION

01

## Objective

Infusing monochrome images with vivid, accurate colors effortlessly.

02

## Exploration Agenda

Redefining automated image enhancement through cGANs.

03

## cGANs

Expertly applying accurate colors by understanding context.

04

## Training cGAN Models

Precision, detail preservation, and visual fidelity ensured in color inference and application.

05

## Applications

Reinvigorating historical photos and enhancing visual appeal.

06

## Transformative Journey

Fusion of technology and artistry bringing grayscale images to vibrant life.

We explored the method of colorization using Conditional GAN (cGANs) proposed by Goodfellow et al.
In deep generative modeling, deep neural networks learn a probability distribution over a given set of data points and generate similar data points.

We used Lab color space instead of RGB to train the models because to train a model for colorization, we should give it a grayscale image and hope that it will make it colorful.

We used a UNet-based architecture for the generator, and used a convolutional PatchGAN for the discriminator.

# Aim of the Project and workflow

# Aim of our project

**Proposed Method**

The cGAN takes a grayscale image as input and generates a corresponding colorized image.

**Training Model**

Teaching machines complex patterns using U-Net architecture and PatchGAN discriminator

**Evaluate**

The Discriminator and GAN loss also figuring out the Generator loss

# Workflow

## Stage 1
### Understanding the context
Studied about different methods for colorization of a grayscale image

## Stage 2
### Retrieving the data
ImageNetmini-1000 which is an image database organized by WordNet hierarchy.

## Stage 3
### Data Preparation
We used Lab color space instead of RGB to train the models.

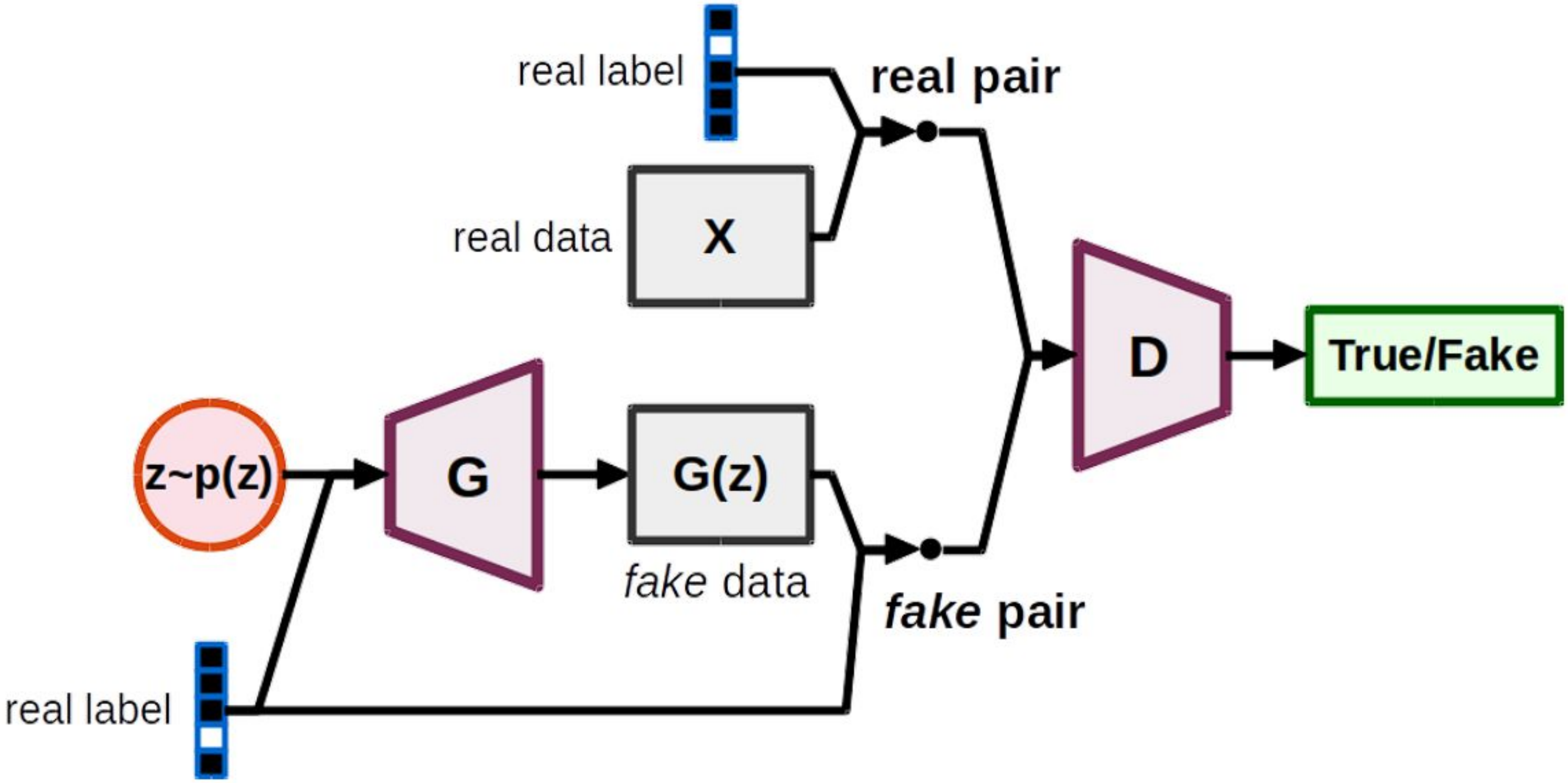
## Stage 4
### Building Models and Training
We built a cGAN and implement U-NET as generator & PatchGAN as discriminator

## Stage 5
### Evaluation of Models & Reporting
As a final result, we got a Discriminator and GAN loss and a Generator loss as well

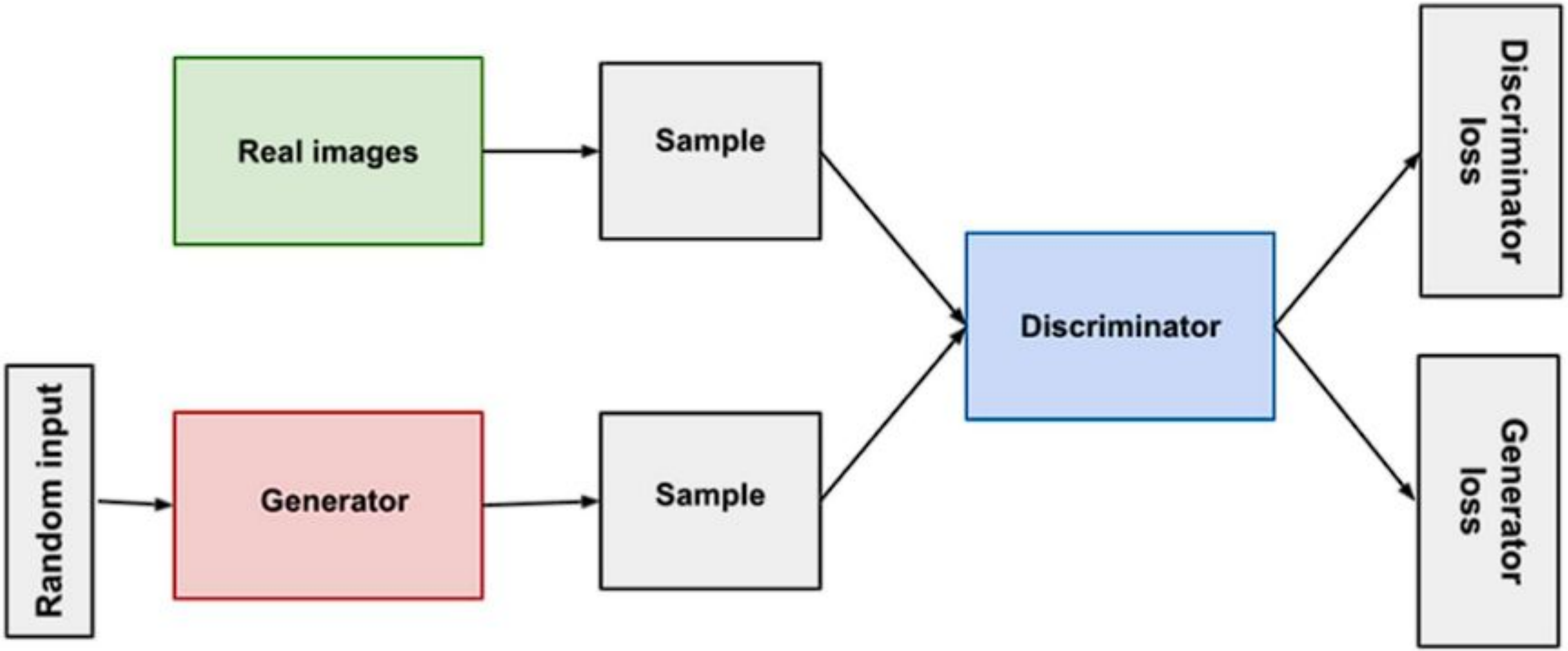
# Prerequisites

Concept 01

# cGAN Architecture



# GAN Architecture



## cGAN

- Incorporates conditional information during training.
- Utilizes additional input data to guide the generation process.
- Enables specific control over the generated outputs based on provided conditions or labels.
- It excel in tasks requiring controlled data generation, like, image-to-image translation, text-to-image synthesis, etc.

**Vs**

## GAN

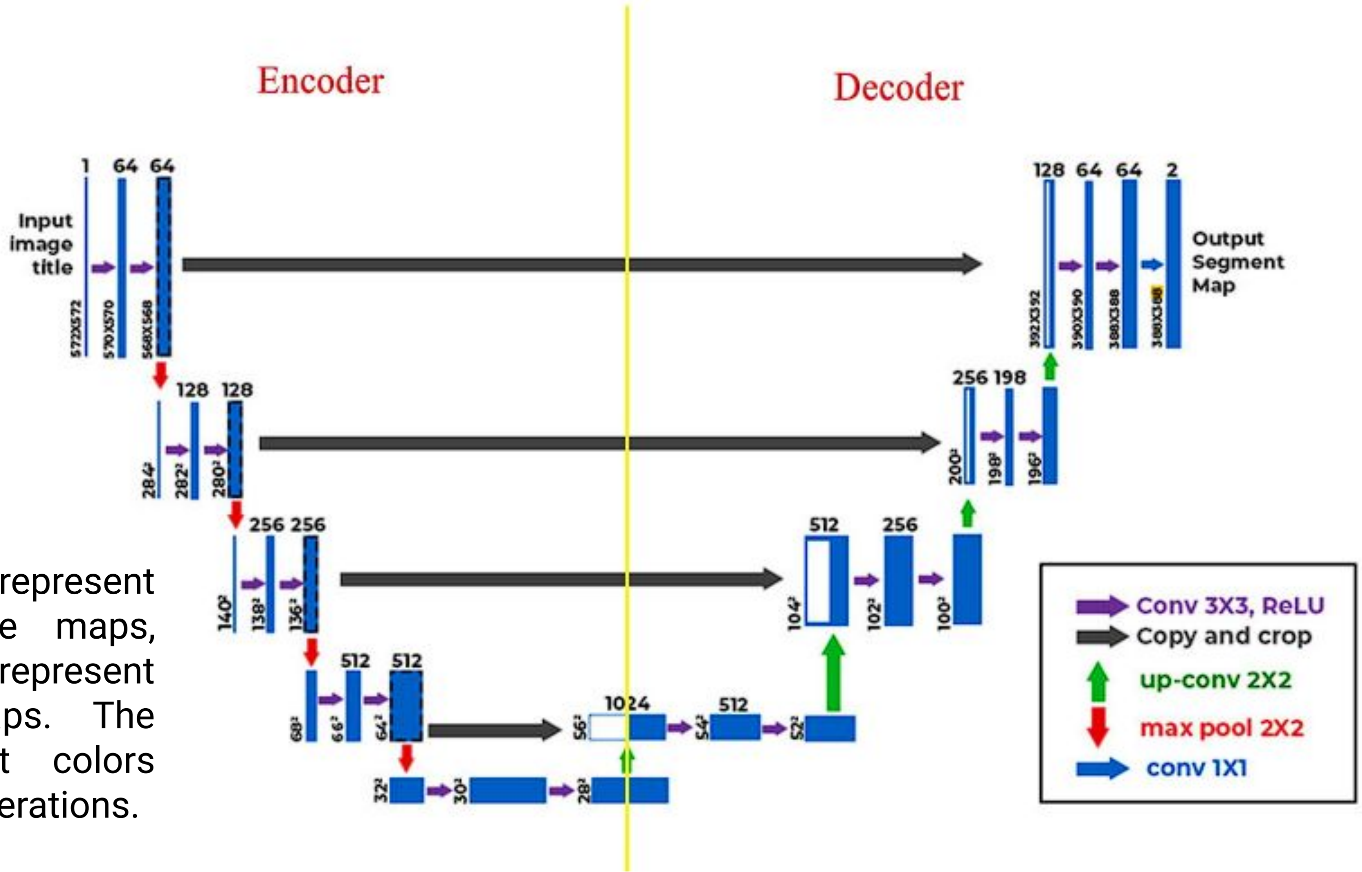
- Unconditional model generating data from random noise.
- Comprises of generator & discriminator in a competitive training setup.
- Learns to generate data without explicit control over the output characteristics.
- GANs find use in unstructured data generation like images, music, text, etc.

## Key difference

GANs generate data purely from noise, lacking control over output specifics. While, cGANs introduce conditional information, allowing targeted generation based on input conditions, labels, or context.

Concept 02

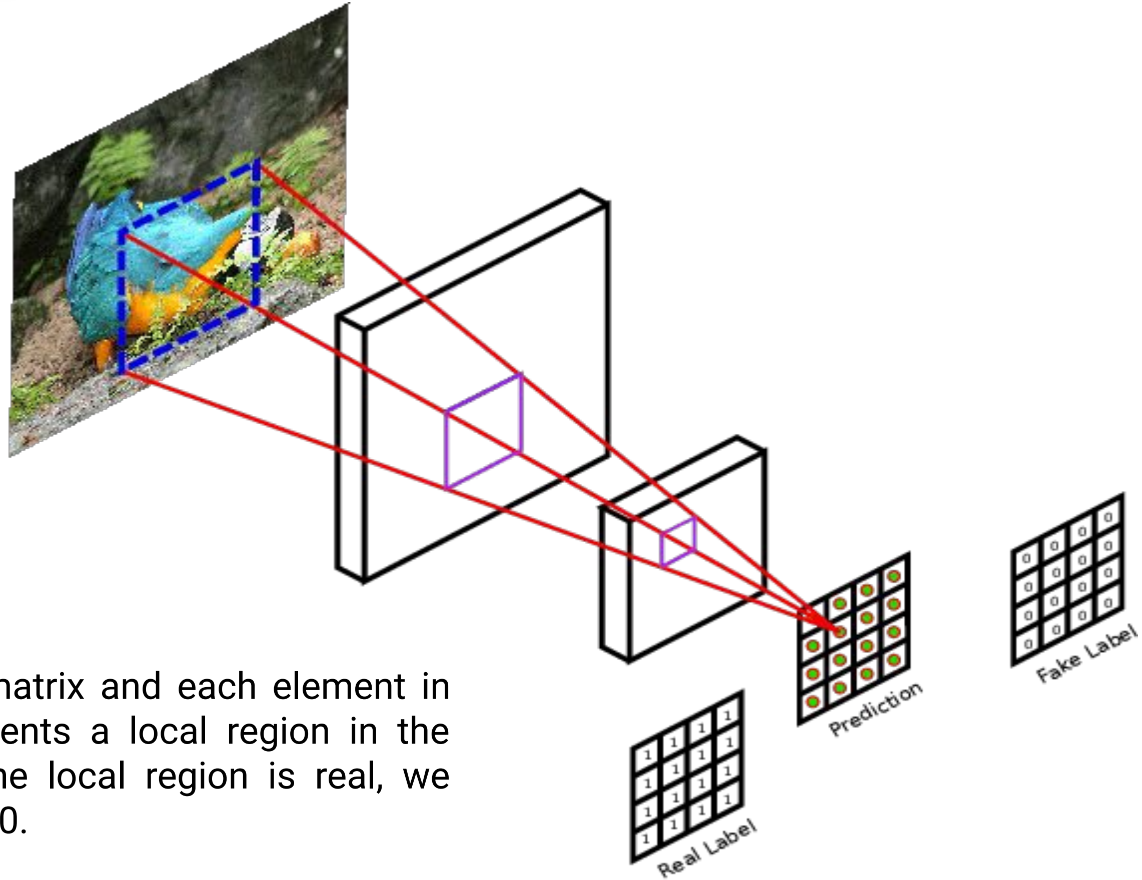
# U-Net Architecture




The blue boxes represent multi-channel feature maps, while white boxes represent copied feature maps. The arrows of different colors represent different operations.

Concept 03

# PatchGAN Discriminator




The output is a matrix and each element in the matrix represents a local region in the input image. If the local region is real, we should get 1, else 0.

Concept 04

# Loss Functions

## Adversarial Loss (GAN Loss)

- The generator loss measures how well the generator is able to fool the discriminator, while the discriminator loss measures how well the discriminator is able to distinguish between real and generated samples.
- Generator loss -
  The generator tries to make its fake stuff look real by minimizing how much the discriminator can tell it's fake.
- Discriminator loss -
  The discriminator loss maximizes accuracy by measuring the binary cross-entropy between actual labels and the discriminator predictions.

## Pixel-wise Loss

- L1 Loss (Mean Absolute Error) –
  This computes the average absolute differences between corresponding pixels of the generated ($G(x)$) and ground truth ($y$) images.
- L2 Loss (Mean Squared Error) -
  This measures the average squared differences between corresponding pixels of the generated ($G(x)$) and ground truth ($y$) images.
- Both L1 and L2 Losses aim to minimize the pixel-wise differences between the generated and ground truth images during training.

# Dataset Description & Transformation

# Dataset Description

## ImageNet Dataset

Contains 10000 images of which we took 5000 images


## Augmentation

Random horizontal flips

Real data

Augmented data

## Training and testing dataset

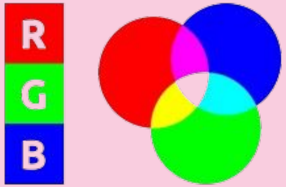
80% Training

20% Testing

# Transformation of Data

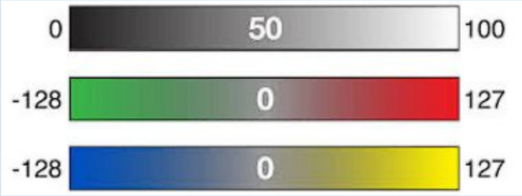
## RGB Color Space

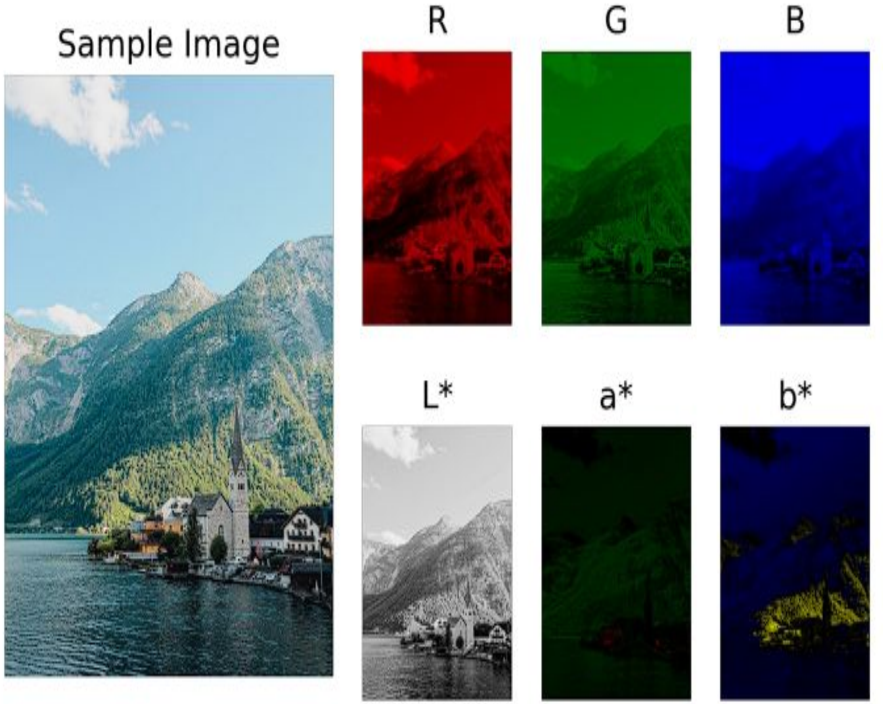
Different channels for red, green, and blue




## L*a*b Color Space

L: Lightening, a: Green to Red, b: Blue to Yellow



# Learning Framework

## 01 Transforming colored images to grayscale

From colored to grayscale using L channel from L*a*b

## 02 Implementing the UNET Autoencoder (generator)

feature extraction and detailed image reconstruction in an integrated architecture.

## 03 Implementing the PatchGAN (Discriminator)

evaluates image patches for better local detail perception

## 04 Model Training

Updating  generator and discriminator adversarially using paired data and specific loss function.

## 05 Analysing the loss function

guides model improvement by measuring prediction accuracy vs actual values.

# Results

# Evaluation for 45 epochs

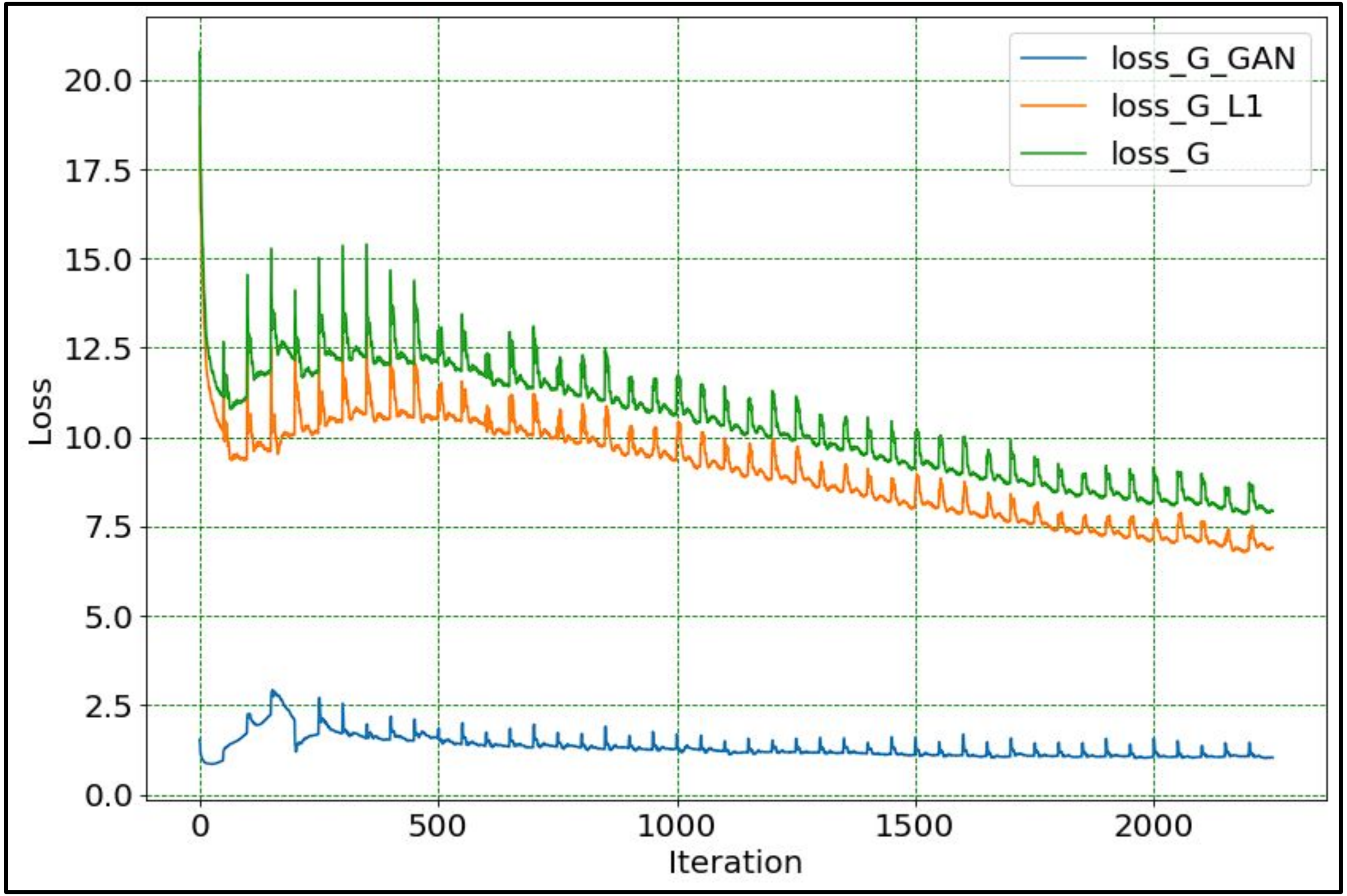

Both the generator and discriminator networks should be trained across multiple epochs to understand the relationship between grayscale and color images. Extending the number of epochs enables the network to grasp more deeper connections, leading to improved colorization accuracy.

Training for more epochs or using additional data typically results in more realistic outcomes. However, due to certain resource constraints, our training was limited to 45 epochs.

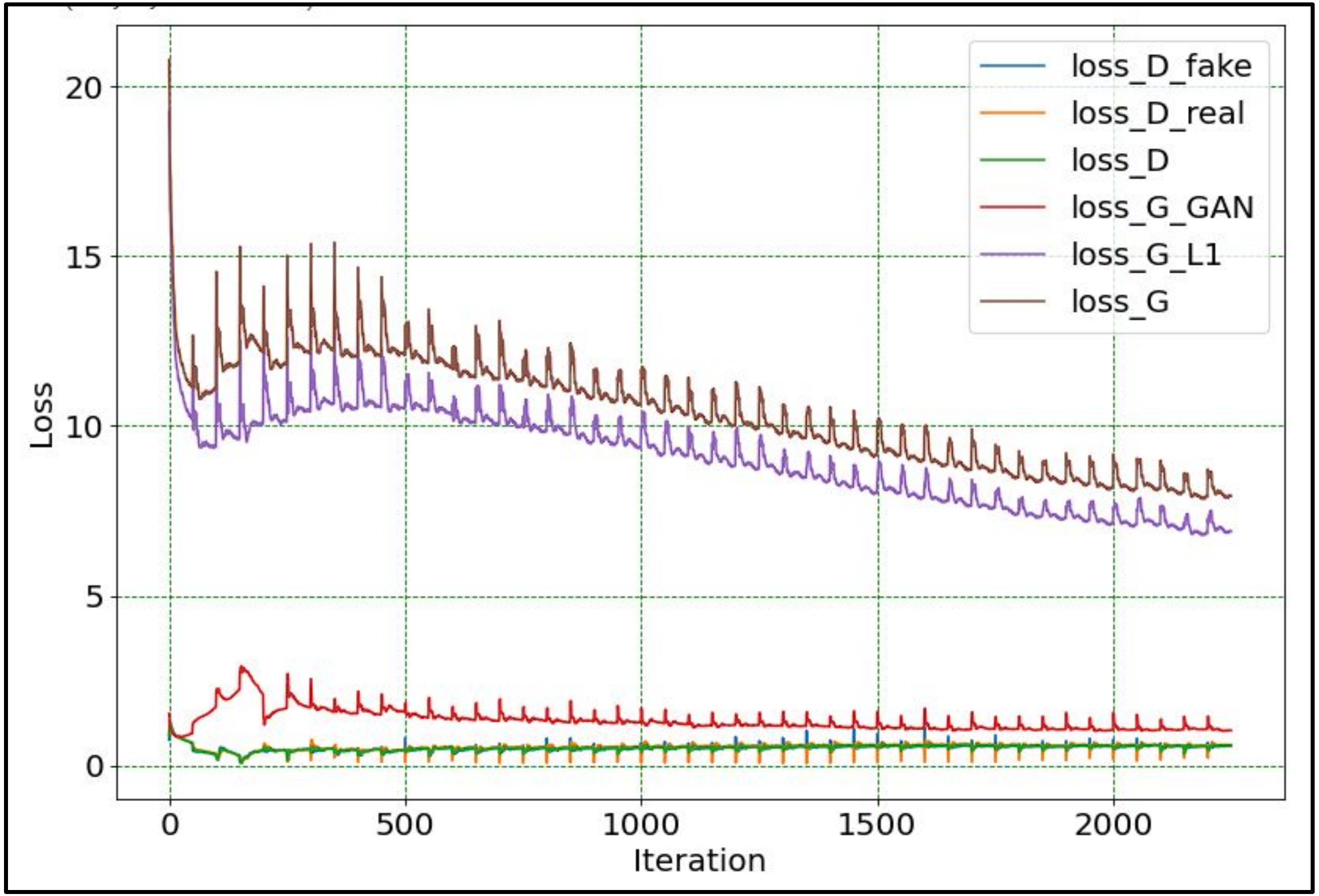
# Results after 45 epochs

By the 45th epoch, the model showed competence in distinguishing and colorizing certain objects. However, it struggled with smaller objects and deeper details, exhibiting limitations in accuracy.

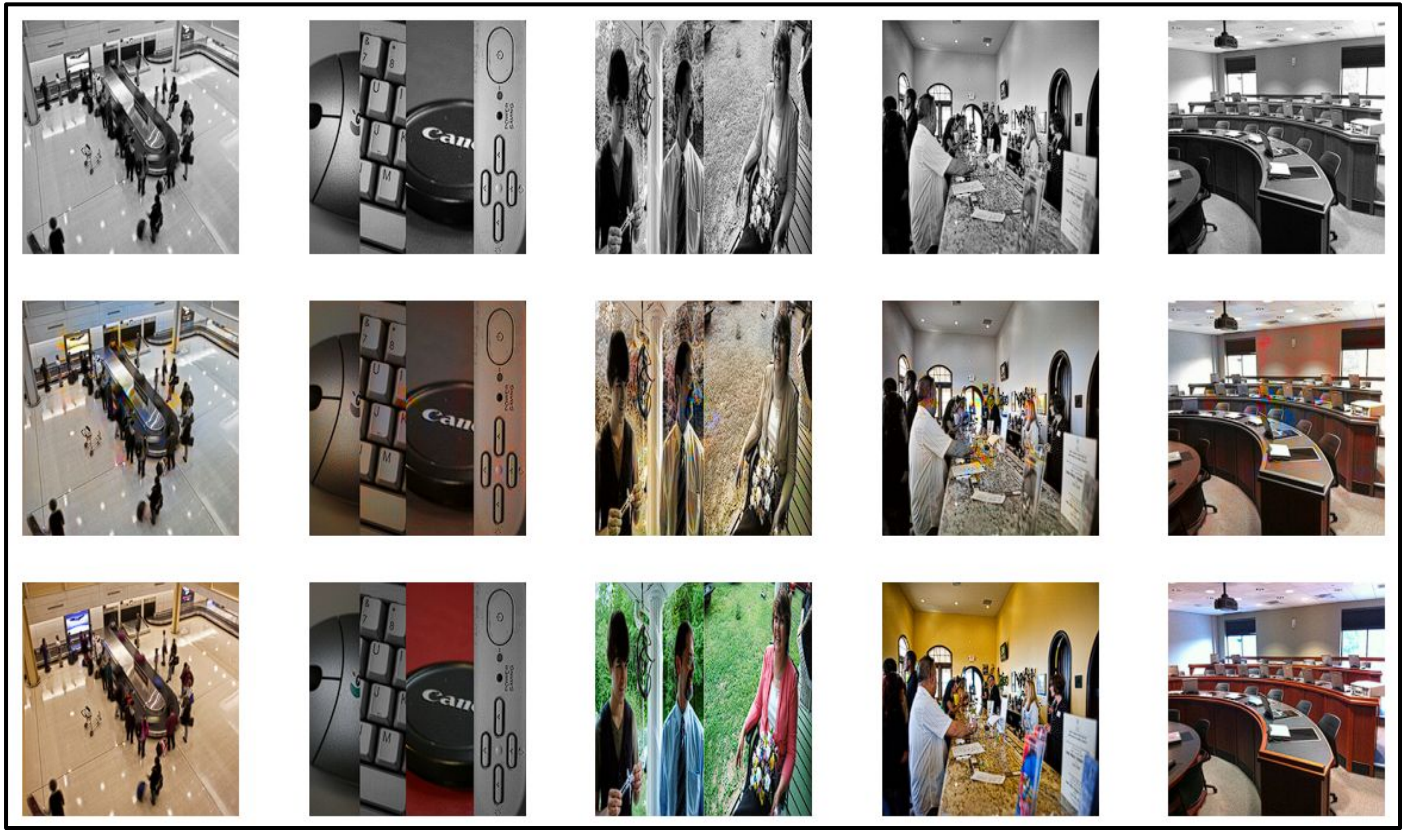
# The Generator loss for 45 epochs



# The Discriminator and G_GAN loss for 45 epochs



# Results after 45 epochs for COCO-2017 dataset (Testing for unseen data)



# Applications

Photography, Historical Archives, Media

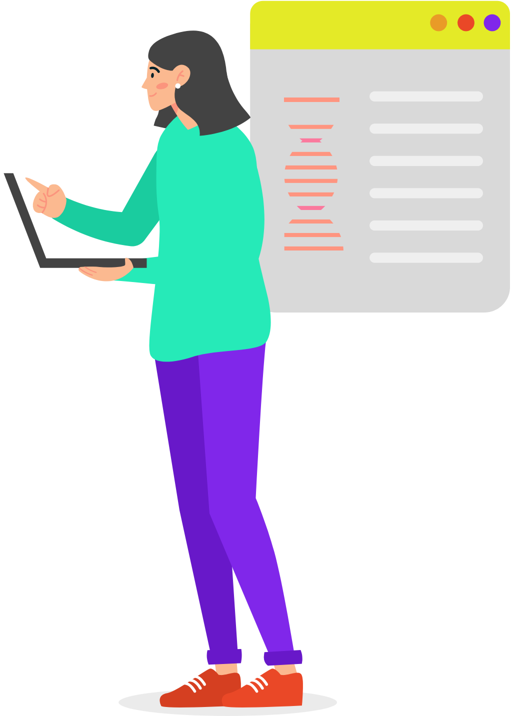
Healthcare, Medical Imaging

Entertainment, Graphic Design

Fashion, Textile Industry

Machine Learning, Computer Vision

Museums, Archives



# Conclusion

cGan has a good performance, even with small dataset.

Also with limited resources we can get reasonable results!

With bigger dataset and a greater number of training epochs the results will be more realistic.

(which demands for more resources)

The capabilities of cGANs in image colorization gave a diverse applications across various domains, including historical photograph restoration, entertainment, medical imaging enhancement, and many more!!