Rapport

Image Colorization using GANs

Realized by : ***Supervised by :***

Academic Year 2022-2023

Contents

Abstract

Our project focuses on the implementation of generative adversarial networks to colorize grayscale images. Unlike traditional GANs that use random probability distributions, we used a Conditional Generative Adversarial Network (cGAN) that takes a specific type of input. Specifically, our cGAN takes a fixed-size grayscale image (32x32 pixels) as input and generates a corresponding colorized image of the same size.

We found that the process of colorizing images that can be effectively represented by a limited set of objects (e.g., animals, vehicles, landscapes) is particularly efficient. By using a cGAN with a fixed input size, we can achieve faster training and better performance in colorizing images. Additionally, we employed several techniques to improve the training process, such as using the LeakyReLU activation function and implementing batch normalization.

Furthermore, our project contributes to the field of computer vision by demonstrating the potential of using generative adversarial networks for colorization tasks. This approach can have broad applications in fields such as medical imaging, satellite imaging, and artistic image rendering. Overall, our project demonstrates the effectiveness of cGANs for grayscale image colorization and provides insights into the optimization of the training process for such networks.

Key Words : Generative adversarial networks(GAN), Conditional Generative Adversarial Network (cGAN), Grayscale images, Colorization, Fixed-size input, Computer vision

1 Introduction

1.1 General

The task of image colorization using GANs is not only about adding colors to grayscale images, but it is also about generating plausible and realistic colorization that aligns with the structure and context of the image. As you mentioned, colors have the ability to drive emotions and influence the aesthetic appeal of an image. In the case of image colorization, the GANs need to understand the context of the image and generate colors that are appropriate for that context. This is a challenging task, as the GANs need to not only understand the relationships between different objects in the image but also the relationships between different colors.

For example, when colorizing a black and white image of a house, the GANs need to understand the structure of the house and generate colors that are appropriate for the different parts of the house such as the walls, roof, doors, and windows. Additionally, the GANs also need to understand the relationships between different colors and how they can be used to create a cohesive and visually pleasing image.

Image colorization using GANs is a challenging task that requires the GANs to understand the context and relationships within the image, in order to generate plausible and realistic colorizations that align with the structure and context of the image.

1.2 Objective

The main objective of this project is to use GANs to color fixed-size black and white images of any object. The solution to this problem can have a wide range of applications, such as automating the colorization of historical black and white photographs, enhancing the aesthetic look of images for UI design, and many more.

In the case of UI developers, the solution can be used to quickly and easily generate different color schemes for their designs, which can be tested and iterated on to determine the color combinations that appeal to users. This can save a lot of time and effort compared to manually selecting colors and can lead to more visually pleasing and user-friendly designs.

Additionally, this solution can also be used in other areas such as interior design, fashion design, and advertising, where color is a crucial element to make the final product visually appealing.

Overall, the solution to this problem statement can have a wide range of applications and can be used to enhance the aesthetic look of various objects and designs.

2 Related works

There has been a significant amount of research in the field of image colorization using GANs. One of the first works on this topic is the paper "Automatic Colorization" by Cheng et al.(2016)[1], which proposed an unsupervised approach for colorizing grayscale images using a GAN. The authors used a variant of GAN called Colorization-GAN, which consists of a generator network that produces a colorized version of the input grayscale image, and a discriminator network that assesses the realism of the generated image. They trained the model on a dataset of color images and showed that it can generate plausible colorizations of grayscale images.

Another notable work is "Perceptual Losses for Real-Time Style Transfer and Super-Resolution" by Johnson et al. (2016)[2], which proposed the use of a pre-trained VGG network as a feature extractor for the generator and discriminator networks in the GAN. This allows the GAN to learn a feature representation that is more closely aligned with human perception, resulting in more realistic colorizations.

In "Deep Video De-Interlacing" by Wang et al.(2018) the authors proposed a GAN-based approach to colorize interlaced video. They use a two-stage training process, where the first stage trains a model to generate plausible interpolated frames, and the second stage trains a model to colorize the interpolated frames. They also used a temporal loss function to ensure consistency between consecutive frames.

In "Deep joint demosaicing and denoising" by Kamal et al.(2019) authors proposed a GAN-based approach for demosaicing and denoising of raw image sensor data. They use a generator network that produces a colorized version of the input grayscale image, and a discriminator network that assesses the realism of the generated image. They trained the model on a dataset of real raw images and showed that it can produce high-quality color images with less noise.

In "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network" by Shi et al. (2016)[3], the authors proposed the use of a sub-pixel convolutional neural network (CNN) as the generator network in the GAN. This allows the GAN to produce high-resolution colorized images in real-time.

Overall, the literature suggests that GANs can be effectively used for image colorization, and that incorporating pre-trained networks and other advanced techniques can lead to improved results. However, there is still room for further research in terms of fine-grained details and preservation of texture in the colorized images.

3 Generative Adversarial Networks

3.1 Architecture

Generative Adversarial Networks (GANs) are a type of deep learning architecture that consists of two main components: a generator network and a discriminator network.

The generator network is responsible for generating new data samples that are similar to the training data. It takes in a random noise vector as input and produces a generated sample as output. The generator network typically consists of several layers of neural networks, such as fully connected layers or transposed convolutional layers. The goal of the generator is to learn a mapping from the noise vector to the training data distribution, so that the generated samples are indistinguishable from the real ones.

The discriminator network is responsible for distinguishing between the generated samples and the real data samples. It takes in a sample (either real or generated) as input and produces a probability as output, indicating how likely the input sample is to be real. The discriminator network typically consists of several layers of neural networks, such as convolutional layers and fully connected layers, and a sigmoid or softmax activation function at the end to produce a probability. The goal of the discriminator is to correctly identify whether a given sample is real or generated.

Both the generator and discriminator networks are trained together in an adversarial manner. The generator is trained to produce samples that are realistic enough to fool the discriminator, while the discriminator is trained to correctly identify the real samples from the generated ones. This process continues until the generator produces samples that are indistinguishable from real ones.

In addition to these two main components, GANs can also include other elements such as auxiliary networks, encoders, or attention mechanisms to improve performance. GANs can be applied in various fields such as image generation, image colorization, image super-resolution, image synthesis, video synthesis, and many more.

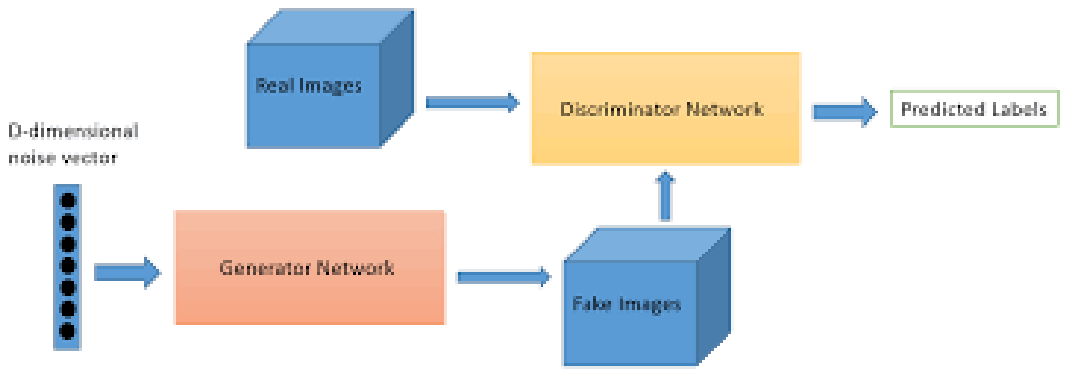


Figure 1: GAN architecture

3.2 Working

Generative Adversarial Networks (GANs) work by training two neural networks, a generator and a discriminator, together in an adversarial manner.

The generator network is trained to produce new data samples that are similar to the training data. It takes in a random noise vector as input and produces a generated sample as output.

The discriminator network is trained to distinguish between the generated samples and the real data samples. It takes in a sample (either real or generated) as input and produces a probability as output, indicating how likely the input sample is to be real.

During training, the generator and discriminator networks are trained in an alternating fashion. The generator produces a batch of fake samples, which are then passed to the discriminator along with a batch of real samples. The discriminator is then trained to distinguish between the real and fake samples.

Next, the generator is trained to produce samples that are more realistic by adjusting its parameters based on the feedback from the discriminator. The generator is updated so that the discriminator's output for the generated samples is closer to 1 (indicating that the generated samples are more realistic). This process continues until the generator produces samples that are indistinguishable from real ones.

The final generator network is able to produce new samples that are similar to the training data. These samples can be used for various tasks such as image generation, image colorization, image super-resolution, video synthesis, and many more.

Overall, GANs work by training a generator network to produce samples that are similar to the training data, and a discriminator network to distinguish between real and generated samples. The generator and discriminator networks are trained together in an adversarial manner, where the generator produces samples that are more realistic based on the feedback from the discriminator.

3.3 Training

3.3.1 Training Discriminator

Training the discriminator network in a Generative Adversarial Network (GAN) involves the following steps:

1. The generator produces a batch of fake samples, which are then combined with a batch of real samples.

2. The discriminator is presented with the combined batch of real and fake samples and is trained to distinguish between them.

3. The discriminator's goal is to correctly classify the real samples as "real" and the fake samples as "fake" by adjusting its parameters.

4. The discriminator's parameters are updated using a optimization algorithm such as stochastic gradient descent (SGD) or Adam, using the backpropagation method to calculate gradients of the network's loss function with respect to the parameters.

5. The process is repeated for a fixed number of iterations or until the discriminator reaches a satisfactory level of performance.

6. The discriminator's performance can be measured using metrics such as accuracy, cross-entropy loss, or precision-recall.

It's important to note that the discriminator is only updated based on the fake samples generated during that iteration, this way the generator can learn from the feedback of the discriminator and improve its ability to generate realistic samples. In addition, the generator is trained to produce samples that are more realistic by adjusting its parameters based on the feedback from the discriminator, this way the generator and discriminator are trained in an adversarial manner.

3.3.2 Training Generator

Training the generator network in a Generative Adversarial Network (GAN) involves the following steps:

1. The generator produces a batch of fake samples, which are then passed to the discriminator. The discriminator assesses the realism of the generated samples and produces a probability score indicating the likelihood that each sample is real.
2. The generator's parameters are updated in a way that maximizes the probability score assigned by the discriminator to the generated samples. This is done by adjusting the generator's parameters to produce samples that are more similar to the real data.
3. The generator's parameters are updated using a optimization algorithm such as stochastic gradient descent (SGD) or Adam. The backpropagation method is used to calculate gradients of the generator's loss function with respect to the parameters.
4. The process is repeated for a fixed number of iterations or until the generator reaches a satisfactory level of performance.
5. The generator's performance can be measured using metrics such as inception score, Fréchet Inception Distance (FID), or precision-recall.

It's important to note that the generator is only updated based on the fake samples it generated during the current iteration. This way the generator can learn from the feedback of the discriminator and improve its ability to generate realistic samples. The generator is trained to produce samples that are more realistic by adjusting its parameters based on the feedback from the discriminator, this way the generator and discriminator are trained in an adversarial manner.

3.3.3 Simultaneous training of Discriminator and Generator

Once both objective functions are defined, they are learned together using alternating gradient descent methods. Modify the parameters of the generator model and perform one iteration of gradient descent on the discriminator using the real and generated images. Then switch sides. Fix the discriminator and train the generator for another one iteration. Train both networks in alternating steps until the generator produces high quality images.

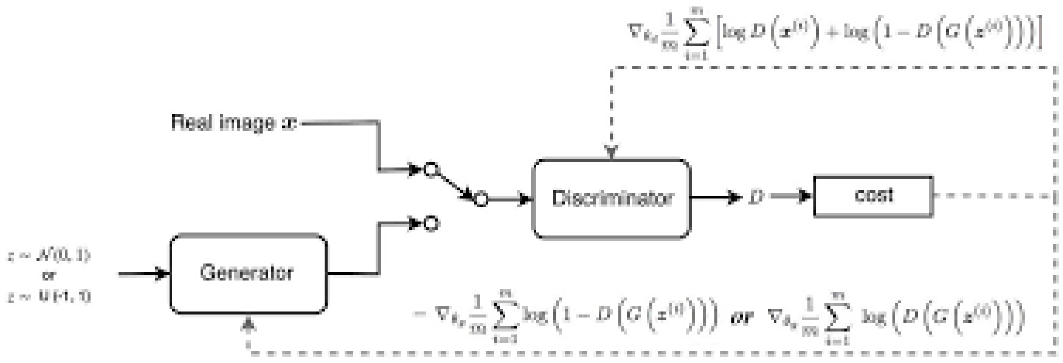

Real image x
Generator
Discriminator
D
cost
$z \sim \mathcal{N}(0,1)$ or $z \sim U(-1,1)$
$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$
$-\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$ or $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(z^{(i)}\right)\right)\right)$

Figure 2: Training GAN

loss function

$$\min_G \max_D E_{x\sim p_{\text{data}}(x)}[\log D(x)] + E_{z\sim p_{\text{z}}(z)}[1 - \log D(G(z))] \tag{1}$$

4 GAN for Image Colorization

In GANs, the input of the generator network consists of noise vector which are randomly generated. But in the task of Image colourisation problem, such GANs will not work as our GAN is not designed to generate images from random vectors but instead add colours to already existing photos having only one channel (Black and white). So the basic task for our GAN is to add three channels (RGB) with relevant intensities of each color channel. Hence, to address this problem, we use a special flavor of GAN called Conditional GANs which accepts grayscale images (with one intensity channel) as input (i.e G(0 z |x),mathematically). The discriminator input is also changed to be compatible with the conditional GANs. Our final cost functions are as follows with above modifications.

$$\min_{\Theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\Theta_G} -E_x[\log(D(G(0_x|x)))] + \lambda||G(0_x|x) - y||_1 \tag{2}$$

$$\min_{\Theta_D} J^{(D)}(\theta_D, \theta_G) = \min_{\Theta_D}(E_y[\log(D(y|x))] + E_z[\log(1 - D(G(0_x|x)|x))] \tag{3}$$

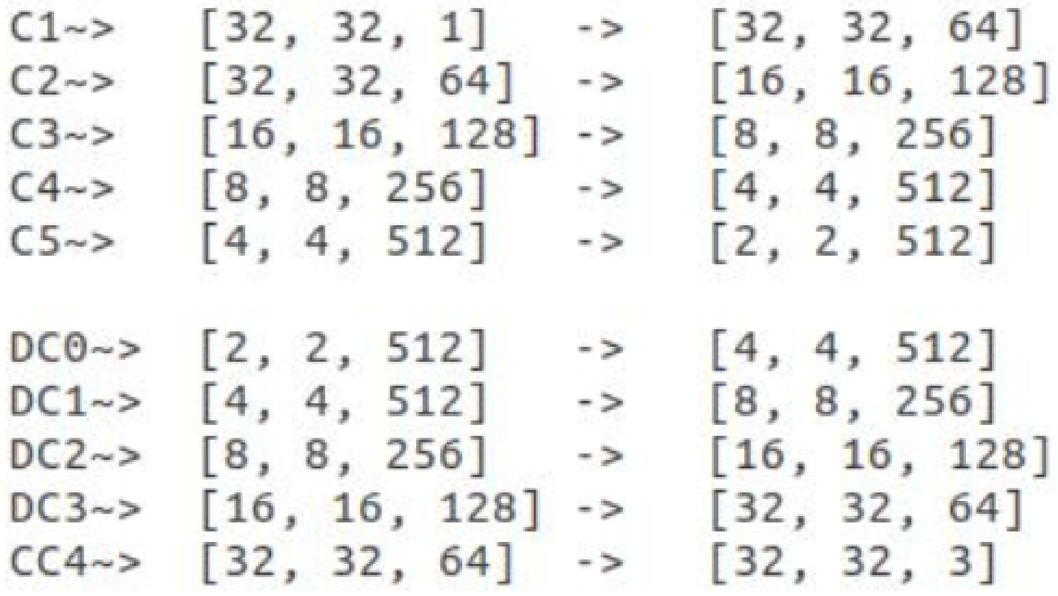
The discriminator takes in input as a pair of grayscale image and generated image, and grayscale image and original image. It then judges for fake or real pair.

4.1 Method

The problem we are trying to solve falls into the category of image-to-image transformation by mapping high-dimensional inputs to high-dimensional outputs. This is actually a pixel-level regression with an output condition that has a structure similar to the input. Therefore, the network must have very similar spatial dimensions of the input and output, and also provide information about the color of each pixel in the original grayscale image.

4.2 GAN Architecture

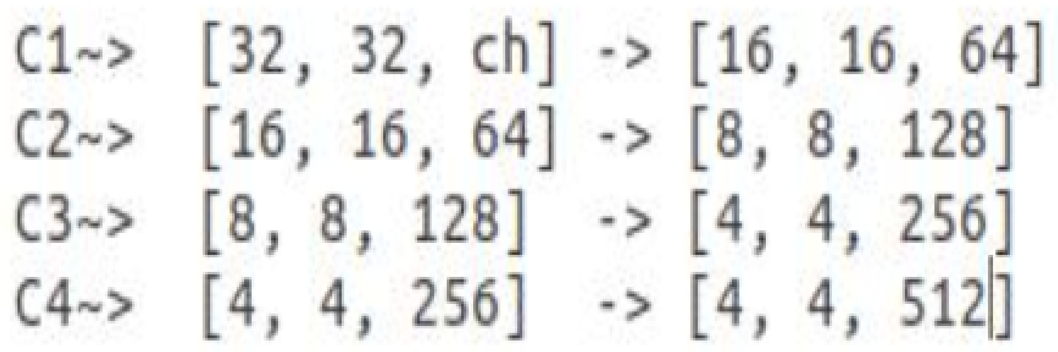
The network for this model is based on "fully connected networks". We use layers of convolutions in Generator. But instead of pooling layers, downsampling of the image is done till it becomes vector of size 2x2 pixels. Then upsampling is done to expand the compressed part and making it to the size of the input sample (i.e 32 x 32 pixels). This strategy is motivated by special types of deep networks called Encoder Decoder networks containing encoding and decoding networks for contracting and then expanding and hence reconstructing the input. This strategy helps in training the network without consuming large amount of memory. The generator takes an input X as a grayscale image having dimensions (32 x 32 x 1). It is initially downsampled with kernel of size 1 and stride 1. After this layer, it is subsequently compressed to image to size (2 x 2) with kernel of size 2 and strides 2. This is done four times after the initial layer, making the matrix of dimensions (2 x 2 x 512). The expansion stages consists of upsampling of the matrix with kernel size 2 and strides 2 except the last layer. Concatenation of i and n-i layers are done to preserve the structural integrity of the image. In the first and second expansive layers, dropout of scale 0.5 is done to introduce noise for robust training of Generator. Batch normalization is done for better training. In our model, we used LeakyReLU with slope of 0.2 as it has shown better performance than ReLU activation function. In the last layer convolution with kernel size 1 and strides 1 is done to construct image of dimension (32 x 32 x 3). "tanh" activation function is used as it has shown to have better performance than linear activation functions. It gives output in the form of matrix containing values from -1 to 1. We train the model to minimize the cosine distance between predicted and the original image. For discriminator, we first concatenate the grayscale image and the predicted or the grayscale image and the ground truth image on the channel axis (axis=3), hence it forms a coloured image. We perform downsampling of the matrix successively using convolutional layer with filter size of (2 x 2) and strides equal to 2. Each layer has Leaky ReLU activation function with slope 0.2 and Batch Normalization is performed at every layer. The last layer is flattened followed by a hidden layer containing 128 units, which are connected to output layer containing 1 unit. The activation function which is used in the last layer is "sigmoid", which gives the probability of the input image to belong to predicted one or the ground truth.



```
C1~>   [32, 32, 1]    ->   [32, 32, 64]
C2~>   [32, 32, 64]   ->   [16, 16, 128]
C3~>   [16, 16, 128]  ->   [8, 8, 256]
C4~>   [8, 8, 256]    ->   [4, 4, 512]
C5~>   [4, 4, 512]    ->   [2, 2, 512]

DC0~>  [2, 2, 512]    ->   [4, 4, 512]
DC1~>  [4, 4, 512]    ->   [8, 8, 256]
DC2~>  [8, 8, 256]    ->   [16, 16, 128]
DC3~>  [16, 16, 128]  ->   [32, 32, 64]
CC4~>  [32, 32, 64]   ->   [32, 32, 3]
```

Figure 3: Generator architecture



```
C1~>  [32, 32, ch] -> [16, 16, 64]
C2~>  [16, 16, 64] -> [8, 8, 128]
C3~>  [8, 8, 128]  -> [4, 4, 256]
C4~>  [4, 4, 256]  -> [4, 4, 512]
```

Figure 4: Discriminator architecture

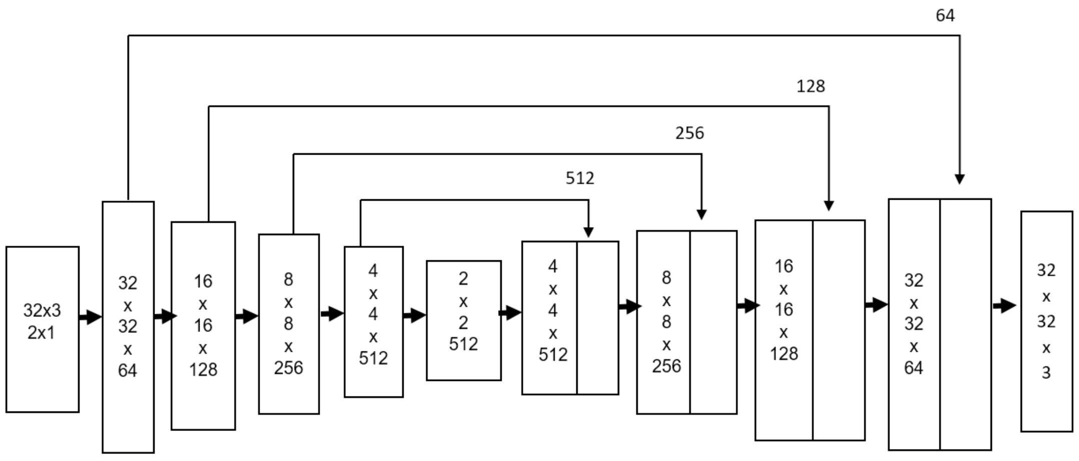


Figure 5: Generator Network visualization

4.3 Training Strategies

For our model, we have used Adam's optimizer. The learning rate for the optimizer is kept to 0.0001. We have made use of open source python libraries Tensorflow and Keras for our model implementation. We have trained the model in free Google Colab GPU. The size of our batch is 50 images.

4.3.1 Batch normalization

The GAN is believed to be hard to train because of a phenomenon called 'mode collapse'. In the model collapse phase, the generator succeeds in fooling the discriminator into believing the same generated output to be true. Hence, the generator generates similar outputs everytime and hence the generation lack variety. This phenomenon is an unwanted state in the training phase of the GANs. To avoid the above phenomenon, we use batch normalization which is proven to reduce the probability of mode collapse. However, batch normalization is avoided in the first layer of the discriminator and generator and the last layer of the generator as suggested by Shravan Murali.

4.3.2 All Convolutional Net

Instead of using spatial pooling, strided convolutions are used.Hence, rather than depending upon fixed downsampling and upsampling, strided convolutions allows model to learn its own upsampling and downsampling. This technique has shown to upgrade the performance of training and helps the network to learn the important invariances with convolution layers only.

4.3.3 LeakyReLU activation Function

LeakyReLU activation function gives better performance then normal ReLU, hence we have used it where ever activation function is used.

4.3.4 Mode Collapse avoidance Strategy

In our case, during some part of the training, we found that the generator generated images using same pattern of colors, along with color grids. This was the outcome of 'mode collapse' as explained earlier. Other than batch normalization, we used a novel approach devised by our friend Kush Jajal, in which we train the generator to avoid using the same colors. This is sort of reverse training for the generator as we intuitively make the generator avoid making the mistake of using the same colors every time.

4.4 SOME Results of the model

Here you see the results of training the model as well as the results of a simple test:

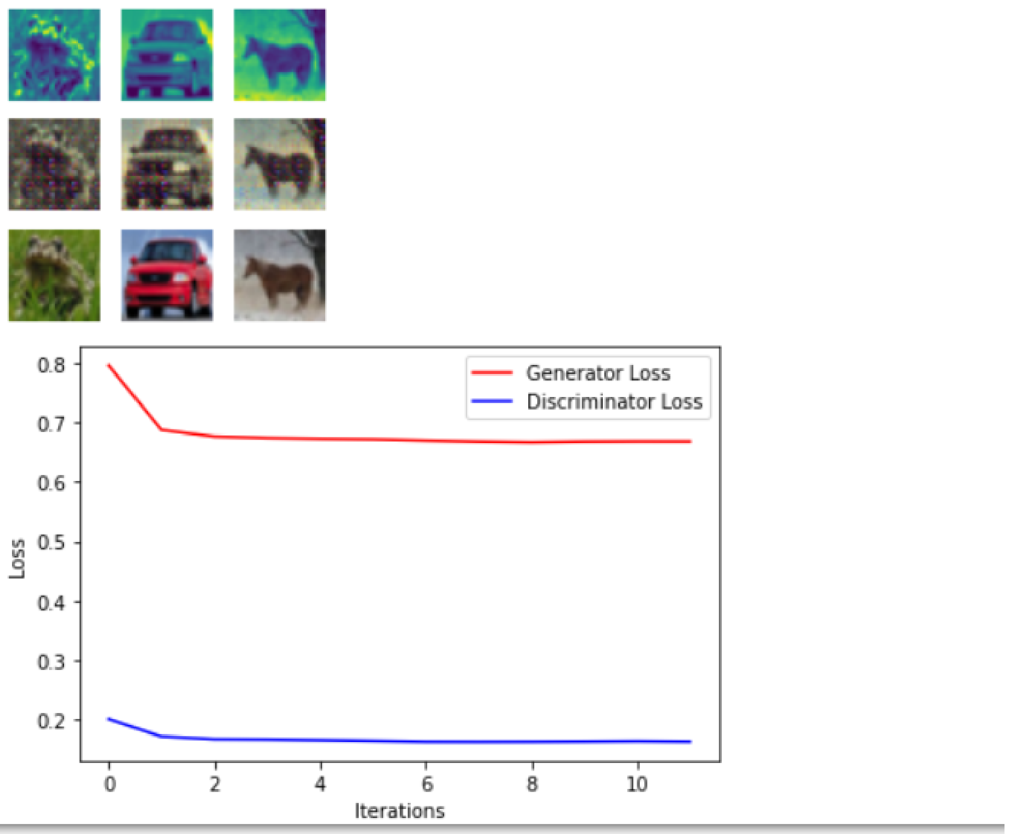
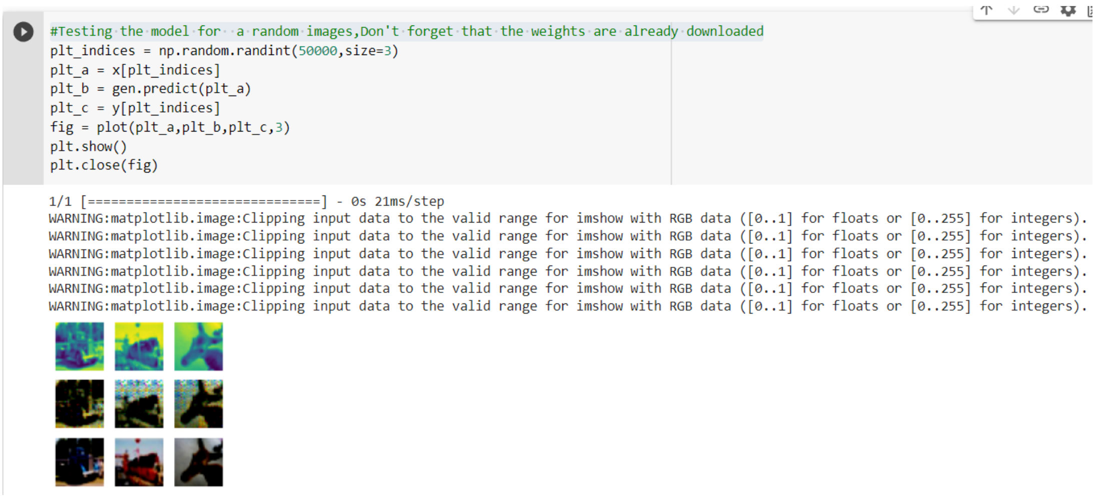


Figure 6: Training results



```
#Testing the model for  a random images,Don't forget that the weights are already downloaded
plt_indices = np.random.randint(50000,size=3)
plt_a = x[plt_indices]
plt_b = gen.predict(plt_a)
plt_c = y[plt_indices]
fig = plot(plt_a,plt_b,plt_c,3)
plt.show()
plt.close(fig)
```

```
1/1 [==============================] - 0s 21ms/step
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Figure 7: Testing results

5 Conclusion

The objective of this project was to explore the use of Generative Adversarial Networks (GAN) for grayscale image colorization. The CIFAR-10 dataset was used for training the model, which contains grayscale images of objects from ten different classes. The primary goal was to produce colorized images that are visually acceptable and similar to the original colored images. Our aim was to investigate the potential of GAN for automatic image colorization, to identify any limitations, and to provide insights into the development of GAN models for future projects.

After training the model, we were able to generate colorized images of the CIFAR-10 dataset that were visually acceptable and similar to the original colored images. However, the model did encounter some issues during the training process. For example, the model sometimes mistook sea water for grass, which resulted in incorrect colorization of images. However, by increasing the number of training epochs, we were able to rectify these issues, resulting in more accurate colorization of images. Moreover, we observed that the model took longer to learn the color red compared to other colors.

This project has successfully demonstrated the potential of GAN for grayscale image colorization. The results showed that GAN can produce colorized images that are visually acceptable and similar to the original colored images, with an acceptable level of accuracy. The challenges encountered during the training process provided insights into the limitations of GAN for automatic image colorization, which could be addressed in future projects. Overall, our work highlights the potential of GAN for transforming various industries and applications, such as image editing and processing, computer vision, and many more.

References

[1] Cheng, Z., Yang, Q., and Sheng, B. Deep colorization. *CoRR abs/1605.00075* (2016).

[2] Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision* (2016), Springer, pp. 694–711.

[3] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 1874–1883.