# BLACK AND WHITE IMAGE COLORIZATION USING GENERATIVE ADVERSARIAL NETWORKS

Umut DEMİREL
191180034
Gazi University
umutdemirel4678@gmail.com

Çağla YÜCE
191180090
Gazi University
cagla38yuce@gmail.com

## ABSTRACT

*Colorization is a popular image-to-image translation problem, and various models, including CNNs and GANs, have been employed to address this issue. For the solution of this problem, we utilized a GAN model with a generator built using the U-Net architecture. Additionally, to facilitate the training of the model, we created a dataset from various sources. We found that the generated dataset accelerated the learning process of the model but constrained its ability to colorize different objects. Furthermore, we identified that the model's color selection often shifted towards blue, mainly due to the prevalent large blue surfaces encountered in the dataset. The report provides numerous examples from the generated images, loss analyses, and details the procedures performed.*

## CCS Concepts

CCS → Computing methodologies → Machine learning → Machine learning approaches → Neural networks

## Keywords

machine learning; deep learning; image processing; neural network; GAN, generative adversarial network; colorization

## 1. INTRODUCTION

Colorizing grayscale images has been a subject of research in machine learning for many years. The reason for this interest lies in the diverse applications of color restoration and image colorization across various fields. Image colorization can be defined as an image-to-image transformation problem that converts a high-dimensional input to a high-dimensional output. The neural network used needs to produce an output with the same spatial dimension as the input and assign color information matching the context to each pixel in the grayscale input image [1].

Different studies and models in the field of image colorization have been examined. The model created in the 'Image Colorization Using Generative Adversarial Networks and Transfer Learning' study was built with the cGAN and PatchGAN structures using the pix2pix architecture. The model, trained with the DIV2K dataset, was evaluated using the Peak Signal-to-Noise Ratio (PSNR) criterion. The proposed method yielded a higher PSNR value compared to other algorithms [2]. In the 'End-to-End Conditional GAN-based Architectures for Image Colourisation' study, the Pix2pix framework was utilized. A U-Net for the generator and a 70x70 PatchGAN for the discriminator were employed. Batch Normalization (BN) and Instance Normalization (IN) processes were combined and applied [3]. The model, trained with samples from the ILSVRC dataset, achieved a PSNR value ranging from 25 to 27 dB, depending on different combinations of BN, IN, and SN [3]. The 'Colorization of Grayscale Images in Deep Learning' study created a model using the Patch-GAN structure and U-Net architecture, trained with the COCO dataset. Due to the high L1 loss, pre-trained weights with the ResNet18 architecture were incorporated, and the Dynamic U-Net module achieved a 78% accuracy over 20 epochs [4]. 'BigColor', unlike traditional GAN-based methods, employs a larger spatial feature map and two control variables instead of BigGAN's spatial coding approach. When tested on the ImageNet-1K validation set, BigColor outperformed other automatic colorization methods [5]. 'RICSPGAN' consists of a noise reduction module 'REDnet' and a colorization module 'SPGAN'. These modules are trained separately and then combined to perform image colorization [6]. DIV2K and ILSVRC2012 datasets are used for the model's training. The model exhibits good perceptual quality, but it fails to show significant improvement in PSNR and SSIM metrics [6].

In this study, we will collect a dataset for image colorization and use generative adversarial networks (GANs) to perform the image colorization process. The dataset used for training will mainly consist of nature and street photos. Selecting similar-content images during the dataset creation process is essential to facilitate the training process. Generative

adversarial networks, as suggested by Goodfellow et al., consist of two small neural networks, a generator, and a discriminator, utilizing unsupervised learning algorithms with supervised loss [7]. The generator neural network will be developed based on the U-Net architecture proposed by Jonathan Long et al [8]. U-Net comprises encoder and decoder blocks and skip connection feature maps are concatenated with encoder blocks, enabling the model to capture both low-level and high-level details [8]. With the solution we are developing, we aim to improve grayscale coloring methods by expanding the coloring process of low-resolution images such as 10x10 to include high-resolution images such as 120x120.

## 2. RELATED WORK

The studies conducted in the field of image colorization using Generative Adversarial Networks are examined under this heading.

### 2.1. Image Colorization Using Generative Adversarial Networks and Transfer Learning

CIELAB color space is a color space defined by the International Commission on Illumination (CIE) in 1976, encompassing all colors perceived by the human eye. In contrast to the RGB color space, there is little correlation between its three components. In this color space, L represents the brightness of the image, A represents the color position between green and red, and B represents the color position between blue and yellow.

In the study titled 'Image Colorization Using Generative Adversarial Networks and Transfer Learning,' which focuses on colorizing grayscale images, the L grayscale input image is used, and the color outputs A and B are predicted and generated by combining the three components of the color image. The conditional Generative Adversarial Network (cGAN) used in the model enables the generation of images conditionally by a generator model [2]. These networks are suitable for transforming input images into output images. The use of cGAN networks establishes a mapping between the two domains. The VGG19 model is employed to ensure that the generalized information found in the top layers of CNN networks is also present in the middle layers but with less sensitivity [2]. The discriminator network is constructed using the pix2pix architecture, incorporating the PatchGAN concept for modeling the structure at high frequencies.

The high-resolution DIV2K dataset from the NTIRE (New Trends in Image Restoration and Enhancement) competitions is utilized for training and testing the model. The Peak Signal-to-Noise Ratio (PSNR) criterion is employed for evaluating the model. The proposed method achieves the highest PSNR value compared to other algorithms [2].

### 2.2. End-to-End Conditional GAN-based Architectures for Image Colourisation

The goal of the study titled "End-to-End Conditional GAN-based Architectures for Image Colourisation" is to perform an automatic CNN-based colorization process for grayscale images. The input grayscale image is represented in the CIE Lab color space with the luminance channel (L) [3]. This image requires the assignment of two corresponding ab color channels, expressed as IRH×W×1, representing the dimensions of H and W in pixels [3]. Regarding GAN architectures, the pix2pix framework uses a U-Net as the generator and a 70x70 PatchGAN as the discriminator, producing output probability maps based on the discrimination of N x N patches in the input field.

Batch Normalization (BN), a commonly used mini-batch normalization technique in deep learning, is crucial for stabilizing the training of GANs [3]. Additionally, Instance Normalization (IN) is employed to accelerate Style Transfer [3]. In this approach, BN and IN are combined and used together. Weight regularization is utilized to enhance the generalization and stability of the network during training. Specifically, the use of sigmoid activations in GAN training can lead to undesirable outcomes, which is addressed by employing Spectral Normalization (SN). In the colorization process, N discriminators at different scales are used to provide sensitivity to small areas and local details [3].

The dataset used for training the model is selected from examples used in the ILSVRC (ImageNet Large Scale Visual Recognition Challenge). The test dataset is created by randomly selecting 10 images per class from the training set. All images are resized to 256x256 pixels and transformed into the CIE La*b color space [3]. The obtained model's Peak Signal-to-Noise Ratio (PSNR) values exhibit variations between 25-27 dB based on the combined or separate use of BN, IN, and SN [3].

### 2.3. Colorization of Grayscale Images in Deep Learning

The study titled "Colorization of Grayscale Images in Deep Learning" is a work on colorizing grayscale images, created based on Generative Adversarial Networks (GANs). GANs are an effective deep learning approach for developing generative models. In these networks, two models named generator and discriminator are trained against each other, starting

from random noise [4]. While the generator aims to learn the data distribution, the discriminator learns to distinguish between generated and real data [4]. The discriminator should provide a high probability for real data and a low probability for fake data. In the proposed model, the generator takes a grayscale photo and generates a two-channel image. The discriminator tries to differentiate between the generated channels and real photos by combining them. Additionally, the Patch-GAN used divides the image into NxN patches, producing a probability matrix for each patch, allowing the discriminator to provide more detailed feedback [4].

The model uses a combined loss function with L1 to generate realistic colored images, aiding the model in predicting accurate colors. Lab color space is employed for image colorization, as it is more suitable than RGB [4]. The U-Net model, with an encoder-decoder structure used to segment images, ensures preservation of details while narrowing and expanding the image.

The COCO (Common Objects in Context) dataset is used to train the model. After training for 50 epochs, the images are resized to 256x256. The discriminator classifies real images with 55% accuracy, but the L1 loss is high [4]. To reduce the loss, the Dynamic U-Net module is applied by loading pretrained weights with a ResNet18 architecture, and the last two layers of the model are removed. The model is pretrained for 20 epochs with a learning rate of 0.002, using the Adam optimizer. The initial accuracy of the obtained model is 65%, reaching 78% after pretraining [4]. The model produces highly successful results for common situations such as landscapes, plants, and humans, but suggests grayscale tones when colorization is uncertain [4].

## 2.4. BigColor: Colorization using a Generative Color Prior for Natural Images

BigColor employs an encoder-generator network architecture. While the encoder predicts a feature map from an input grayscale image, the generator synthesizes a colored image from this feature [5]. Unlike traditional GAN-based colorization methods, BigColor does not rely on the spatially distributed code of BigGAN. Instead, it utilizes a spatial feature map with a larger dimension. Additionally, it employs two control variables for conditioning both the encoder and generator [5]. The feature map predicted by the encoder has a spatial resolution of 16x16 for a 256x256 resolution input image. The design of the BigGAN was designed using the reverse of the generator [5].

The model is trained by jointly training the encoder, generator, and discriminator using a loss function that incorporates three different terms during training. The model is trained using 1.2 million colored images from the ImageNet training set resized to 256x256. Adam optimizer is used for training, and the training is conducted for 12 epochs. BigColor's effectiveness is evaluated on the ImageNet-1K validation set, demonstrating superior results compared to other automatic colorization methods [5].

## 2.5. Robust Image Colorization using Self Attention based Progressive Generative Adversarial Network

RICSPGAN consists of two modules: the noise reduction module REDnet and the colorization module SPGAN. These modules are initially trained separately and then combined to perform a robust image colorization process [6]. REDNet is employed to remove distortions from grayscale images, utilizing the encoder for feature extraction and the decoder for detail restoration. Self-Attention GAN is included in the model as an alternative to traditional convolution processes, capturing long-range dependencies. Spectral normalization in the model helps stabilize the training of GANs.

The training process progresses by increasing the input image size during training instead of enlarging the U-Net architecture, thereby enhancing generalization ability, and speeding up the training process [6]. The model's training occurs in two stages. In the first stage, a grayscale image is corrupted with Gaussian noise and directed to RedNet-10, producing a denoised image. In the second stage, the denoised image from RedNet-10 is directed to the Self-Attention-based Progressive GAN, equipped with a feature extraction block pre-trained with ResNet34, for colorization.

The model is trained using the DIV2K and ILSVRC2012 datasets. Some colorized outputs of grayscale images from the DIV2K dataset are presented. The perceptual quality of the model is good, although it has not demonstrated significant improvement in some metrics (PSNR and SSIM) [6]. While generative architectures may yield visually pleasing results, they might exhibit some weaknesses in metrics such as PSNR and SSIM [6].

# 3. METHODOLOGY

## 3.1. Dataset

### 3.1.1. Data Collection and Cleaning

The dataset used for training and testing the model was created by combining various sources. The preparation of the dataset was based on the MIRFLICKR dataset, resulting in a total of 3000 RGB images [9]. Images were selected predominantly from nature and street photography, sourced from different repositories. However, the dataset also includes a notable proportion of objects such as humans, animals, and flowers. During the selection process, all pre-existing grayscale images and those lacking sufficient context were extensively filtered out to ensure high quality. A subset of these images was retained to prevent overfitting of the model.

Figure 1. Example images from the dataset

### 3.1.2. Data Augmentation

Since the created dataset includes images obtained from various sources, preprocessing of the data is necessary. Therefore, images obtained from the dataset are initially resized. The input size for the models used in the project was chosen to have a resolution of 120x120. Hence, all images were resized to have a resolution of 120x120. Grayscale versions of these resized images were also generated. Subsequently, normalization was applied to both RGB and grayscale images. Due to the different sources of images in the dataset, they have different naming conventions. This situation leads to some images being included only in the test dataset. To address this, the images were subjected to a shuffle process. Since we have both RGB and grayscale images, each RGB image was paired with its grayscale version to form tuples, and the shuffle process was applied. This way, the dataset was shuffled, and the relationship between RGB and grayscale images was preserved.

The dataset was split into 90% for training and 10% for testing. Finally, both the training and test datasets were transformed into TensorFlow datasets consisting of tuples of RGB and grayscale images. The batch size for dataset usage was set to 64, and prefetching of 32 images per batch was implemented to enhance performance.

## 3.2. Architecture

Generative Adversarial Networks (GANs) are unsupervised learning algorithms that use supervised loss as part of the training. GANs consist of two small neural networks: a generator and a discriminator. Since the project involves the colorization of black and white images, which is an image processing task, both the generator and discriminator are constructed as Convolutional Neural Networks (CNNs). The generator is tasked with producing outputs that cannot be easily distinguished from real data [10]. Generators typically take random noise as input and transform it into complex data examples [10].

On the other hand, the discriminator is responsible for classifying whether a given data is generated by the generator or is real data [10]. Discriminators function as binary classifiers with this capability [10]. In essence, the generator tries to deceive the discriminator by producing outputs that resemble real data [10]. Therefore, GANs leverage the adversarial training method to generate outputs that are close to reality. Both models, the generator, and the discriminator, are trained simultaneously until the generator achieves the ability to produce outputs convincing enough to deceive the discriminator. At the end of a successful training process, the generator will have the capability to generate highly realistic synthetic data [10].

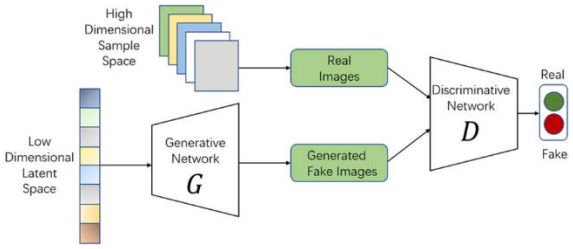


Figure 2. GAN architecture

Z represents the noise input taken by the generator, where the generator is denoted by G(z; θG) using the mapping function [1]. X is an RGB image, and the

discriminator is represented by D(x; θD) using the mapping function, producing a scalar value between 0 and 1 [1]. The generator is trained to minimize the probability of the discriminator making correct predictions, while the discriminator is trained to maximize the probability of making correct predictions. The loss functions used for GAN training are provided below. Typically, it is treated as a min-max problem with a value function V(G, D) [1].

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} E_z\,[\log(1 - D(G(z)))], \quad (1)$$

$$\max_{\theta_G} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_G} E_X[\log(D(x))] + E_z[\log(1 - D(G(z)))]. \quad (2)$$

The generator model developed within the scope of the project is based on the U-Net architecture. The U-Net architecture is a neural network design consisting of interconnected encoder and decoder blocks bridged by a central skip connection, forming a U-shaped structure [8]. The encoder block reduces spatial dimensions while increasing feature channels, whereas the decoder block increases spatial dimensions while reducing feature channels [8]. Each encoder block typically comprises two 3x3 convolutional layers with a ReLU activation function. Following the two convolutional layers, there is a max-pooling layer responsible for reducing spatial dimensions [8]. Each decoder block usually starts with a 2x2 transpose convolutional layer. Subsequently, it is concatenated with a skip connection feature map matching itself from the encoder block. This prevents the loss of features that may occur within the neural network. The decoder block is then completed using two 3x3 convolutional layers. The layer connecting the encoder and decoder is referred to as the bridge and generally consists of two 3x3 convolutional layers [8]. The skip connections in U-Net serve as shortcut connections that indirectly facilitate the flow of gradients to previous layers without degradation [8].

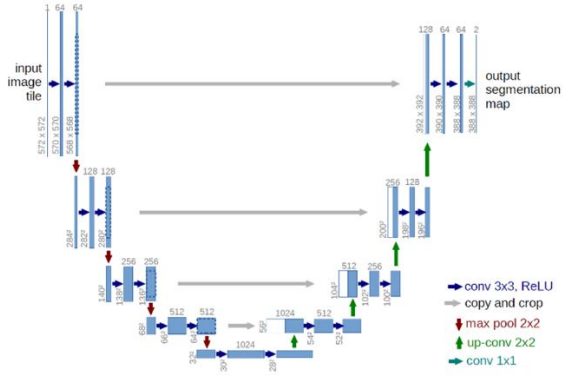


Figure 3. U-Net architecture

The generator model developed within the project encompasses an input layer with a shape of 120x120x1x64. This is due to the generator input consisting of 64 grayscale images with dimensions of 120x120. Three encoder blocks follow the input layer. Encoder blocks, with a LeakyReLU activation function, consist of one 5x5 and two 3x3 convolutional layers and perform feature extraction. A bottleneck layer is present that connects the encoder and decoder. This layer, comprising a single 3x3 convolutional layer, uses the tanh activation function to compress the encoded information into a lower-dimensional representation, producing values in the [-1,1] range.

The bottleneck layer is followed by three decoder blocks responsible for upsampling. Decoder blocks are initially concatenated with skip connection feature maps matching themselves from the encoder block. These concatenation layers assist the generator in capturing both low-level and high-level details by combining features from different stages of the encoder [8]. Following this, the bottleneck layer is succeeded by two 3x3 and one 5x5 convolutional layers. The output generated by the last decoder block is considered the output layer and has a shape of 120x120x3x64. This is because the generator output is a 64 RGB image with dimensions of 120x120.

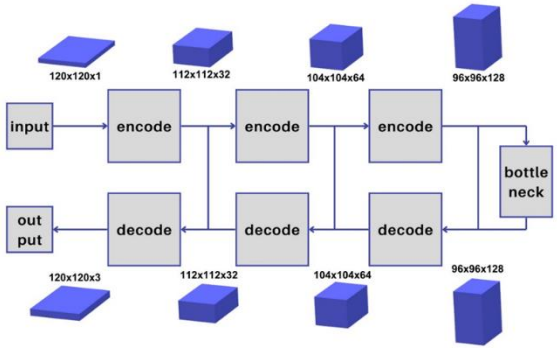


Figure 4. Generator architecture designed within the scope of the project

The discriminator model developed within the project includes an input layer with a shape of 120x120x3x64. This is because the discriminator input consists of 64 RGB images with dimensions of 120x120. Following the input layer, there are four encoder blocks designed to reduce spatial resolution and perform feature extraction. These blocks consist of 2 nxn convolutional layers and a max-pooling layer each. After these four blocks, there is a flatten layer used to reduce the 3-dimensional input to 1 dimension, preparing it for fully connected layers. The output, reduced to 1 dimension, is sequentially passed through four fully connected layers, ultimately reducing it to a scalar output. The last fully connected layer has a sigmoid activation

function, and its output is considered the output layer. If the output of the sigmoid activation function is 1, the image is considered real; if it is 0, the image is considered synthetic. In other words, the discriminator serves as a binary classifier.

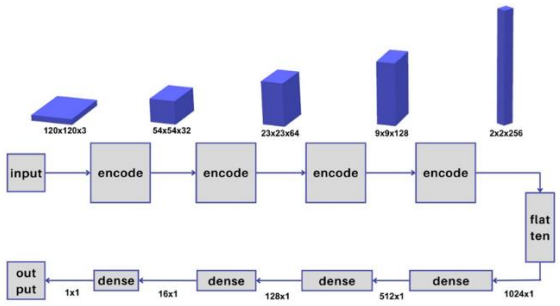


Figure 5. Discriminator architecture designed within the scope of the project

## 4. EXPERIMENTAL DESIGN

Many existing neural networks use a loss function that tends to converge to zero. However, GANs consist of two separate neural networks called generator and discriminator, requiring the definition of two distinct cost functions [10]. The training of GANs is accomplished by striking a balance between these two cost functions [1]. For example, as the generator starts producing more realistic synthetic results, the generator loss will decrease. However, these realistic results confuse the discriminator, increasing its loss. If the model balance is disrupted, both cost functions start oscillating, leading to an occurrence known as non-convergence or mode collapse [1].

For the training of the generator and discriminator models created within the scope of the project, the Adam optimizer was employed. The step size for training was set to 0.0005. Mean squared error loss function was chosen for the generator loss. The generator loss is calculated based on the difference between the synthetic RGB image generated by the generator and the real RGB image. Binary cross-entropy loss function was chosen for the discriminator loss. The reason for selecting binary cross-entropy loss function is that the discriminator model acts as a binary classifier. Since the images are normalized, all values will be between 0 and 1. For this reason, binary cross entropy can be preferred as the loss function of both models if desired. Discriminator loss is calculated based on the outputs obtained by passing both the synthetic RGB image generated by the generator and the real RGB image through the discriminator. The discriminator should produce an output of 1 for real RGB images and 0 for synthetic RGB images. Therefore, the outputs generated for real RGB images are fed into the loss function with a matrix of 1s of the same size, while the outputs generated for synthetic RGB images are fed into the loss function with a matrix of 0s of the same size. The discriminator loss is calculated by summing these two loss values. Before calculating the loss value, random noise is added to the matrices of 1s and 0s to slow down the learning rate of the discriminator.

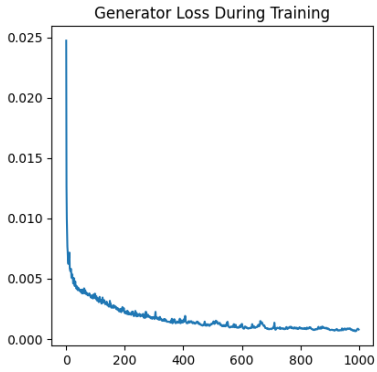


Figure 6. Generator loss during training

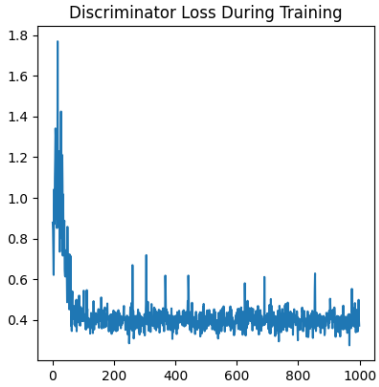


Figure 7. Discriminator loss during training

The generator and discriminator loss values obtained during the training of the GAN model are shown in Table 1.

Table 1. Generator and discriminator losses during training

| Epoch | Generator Error (Mean Squared Error) | Discriminator Error (Binary Cross Entropy) |
|---|---|---|
| **20** | 0.0075 | 0.9756 |
| **50** | 0.0062 | 0.6844 |
| **100** | 0.0056 | 0.4423 |
| **150** | 0.0052 | 0.4255 |
| **250** | 0.0037 | 0.4087 |
| **500** | 0.0024 | 0.3935 |
| **750** | 0.0015 | 0.3980 |
| **1000** | 0.0013 | 0.4002 |

## 5. EXPERIMENTAL RESULTS

The generated GAN model has been trained on the training dataset for approximately 1000 epochs. Following the training, 0.0013 generator loss and 0.4002 discriminator loss values were achieved on the test dataset. Upon examining the GAN outputs, it was observed that the images closely resembled real RGB images, with some photos almost indistinguishable from real RGB visuals. This indicates a successful training process. The presence of imperfections in the outputs is attributed to insufficient training, and it is believed that these issues will be rectified with further training over an adequate duration.

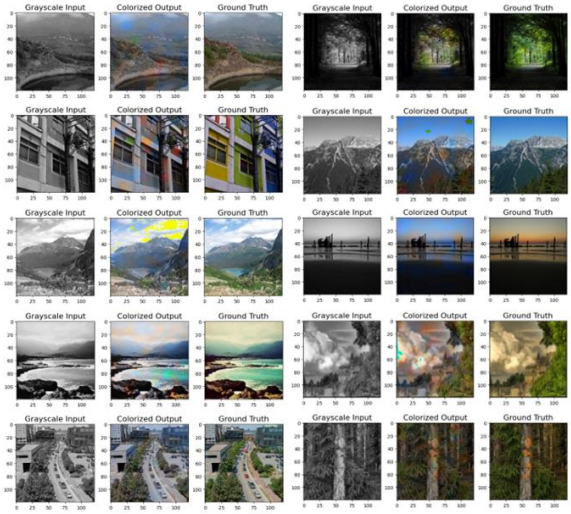


Figure 8. Model output examples on test data set

When the RGB outputs are examined in detail, it is observed that the blue color is more dominant compared to other colors. This phenomenon stems from the large amount of nature and street photos in the dataset. The prevalence of amorphous areas such as the sky, sea, river, and snow in these types of photos, where a significant portion consists of such amorphous regions, is the primary reason for the model's tendency towards blue.

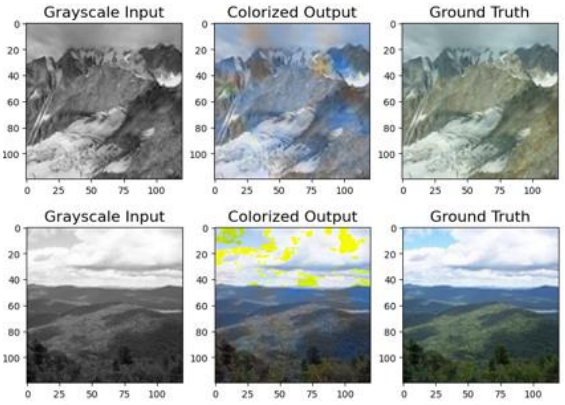


Figure 9. Blue dominance example

Upon close examination of the RGB outputs, it is observed that water areas with original greenish, turquoise, and navy colors produce outputs in the same color. This is due to the GAN model's tendency to color objects in the most frequently encountered colors. This is a common disadvantage of GANs.

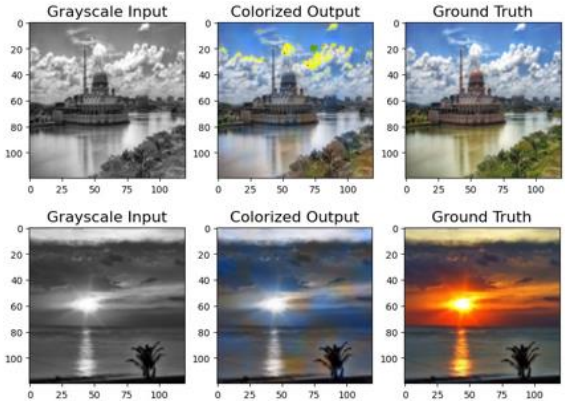


Figure 10. Most frequently encountered colors example

When the RGB outputs are examined in detail, it is noticed that objects such as animals and flowers are not sufficiently colored or are incorrectly colored. This issue arises from the insufficient presence of these objects in the dataset. Therefore, the GAN model fails to learn these objects adequately.

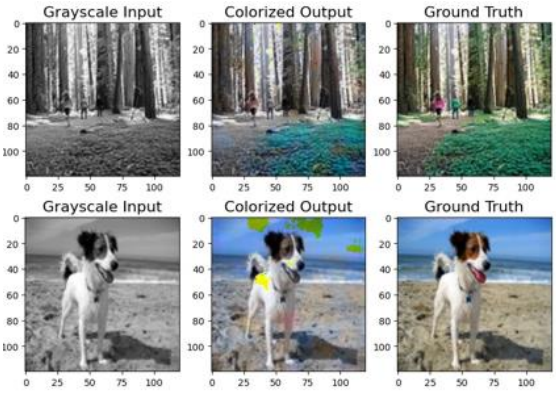


Figure 11. Insufficient presence of objects example

The trained GAN model has been tested on the test dataset, and the obtained loss values are presented in Table 2.

Table 2. Generator losses on the test dataset

| Test Batch | Loss Value (Mean Squared Error) |
|---|---|
| **Batch 1** | 0.00231 |
| **Batch 2** | 0.00279 |
| **Batch 3** | 0.00266 |
| **Batch 4** | 0.00280 |
| **Batch 5** | 0.00383 |
| **Weighted Mean** | **0.00281** |

## 6. CONCLUSIONS

In this study, we successfully generated satisfactory color images from grayscale images using a GAN model. Although the resolution of the images used in the model's training is relatively high (120x120), we believe that selecting images based on their content during the creation of the dataset facilitated the model's learning process. The dominance of the blue color in the generated color outputs is thought to originate from amorphous areas such as the sky and sea, which cover large areas in the dataset and are frequently encountered. It is believed that utilizing an effective boosting algorithm in the generation of color outputs will reduce this dominance. To enable the model to better learn objects that it cannot color sufficiently, images containing these objects can be added to the dataset. Finally, for measuring the model's performance, additional quantitative metrics such as peak signal-to-noise ratio (PSNR) can be employed to provide a detailed performance analysis. The project we have developed will enable people to colorize black and white photos taken in the past. We believe that the project will contribute to humanity both spiritually and nostalgically.

## 7. BIBLIOGRAPHY


[1] Nazeri, K., Ng, E., & Ebrahimi, M. (2018). Image colorization using generative adversarial networks. Articulated Motion and Deformable Objects: 10th International Conference, pages 85-94. doi:10.1007/978-3-319-94544-6_9

[2] Kiani, L., Saeed, M., & Nezamabadi-Pour, H. (2020). Image Colorization Using Generative Adversarial Networks and Transfer Learning. 2020 International Conference on Machine Vision and Image Processing (MVIP), pages 1-6. doi:10.1109/MVIP49855.2020.9116882

[3] Blanch, M. G., Mrak, M., Smeaton, A. F., & O'Connor, N. E. (2019). End-to-End Conditional GAN-based Architectures for Image Colourisation. 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), pages 1-6. doi:10.1109/MMSP.2019.8901712

[4] Pyngrope, A. D., & Kumar, P. (2022). Colorization Of Grayscale Images in Deep Learning. 2022 International Journal of Engineering Applied Sciences and Technology, 11(6), pages 203-212.

[5] Kim, G., Kang, K., Kim, S., Lee, H., Kim, S., Kim, J, Baek, S.H., Cho, S. (2022). Bigcolor: colorization using a generative color prior for natural images. European Conference on Computer Vision, pages 350-366. doi:10.1007/978-3-031-20071-7_21

[6] Sharma, M., Makwana, M., Upadhyay, A., Singh, A., Badhwar, A., Trivedi, A., Saini, A., Chaudhury, S. (2019). Robust Image Colorization Using Self Attention Based Progressive Generative Adversarial Network. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 2188-2196.

[7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems (27), pages 139-144. doi:10.1145/3422622

[8] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3431-3440. doi:10.1109/CVPR.2015.7298965

[9] Huiskes, M., Thomee, B., & Lew, M. (2021). MIRFLICKR Download. MIRFLICKR: https://press.liacs.nl/mirflickr/mirdownload.html

[10] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. IEEE signal processing magazine, 35(1), pages 53-65. doi:10.1109/MSP.2017.2765202