“Image-to-Image Translation (Recoloring) with WGAN”

Anil Ozfirat†, Berker Senol†, Ismail Deha Kose†

***Abstract*—This paper explores the application of deep learning models, specifically Vanilla Generative Adversarial Networks (GANs) and Wasserstein-GAN (WGAN), for the task of colorizing grayscale images. The study aims to enhance the existing image-to-image translation approach by incorporating the WGAN method and evaluating the result using FID, IS, PSNR, and SSIM metrics. The U-Net architecture is utilized as the generator, while the PatchGAN discriminator is implemented to assess the input images at a detailed level. Preprocessing steps, such as resizing and color space conversion, are performed on the dataset to optimize model performance. The combined loss function GAN, L1, and WGAN contribute to generating vivid and accurate images. The proposed approach demonstrates the potential for various applications, including historical photograph restoration and medical image enhancement.**

***Index Terms*—Unsupervised Learning, Deep Learning, Generative Adversarial Network (GAN), Neural Networks, Wasserstein Generative Adversarial Network (WGAN), Image Recoloring.**

I. Introduction

Color have an important place in our lives. Each color hides a different information in it. To give a simple example, traffic lights actually tell us what have to do with itself colors. Machine learning and deep learning techniques continue to be developed so that humanity can access more information.

In recent years, studies on the utilization of deep learning in computer vision and image processing have been increasing visibly. In these studies, One of the most efficient methods employed in these studies is the ”image-to-image translation” approach [1]. In this method, provided with an input image to the model and it is expected to produce an output from this image in line with the requests. One of the most common applications of this method is the colorization of grayscale (black-white) images [2].

Grayscale images can result in certain information loss compared to color images. Due to the absence of color information, fine details in grayscale images may be less pronounced compared to their color images. As a result, grayscale images may lose some level of detail and visual richness when compared to color images. Color is a significant factor, that emphasizes an object’s surface characteristics, depth, and uniqueness. Colorization plays a crucial role in various application domains, such as revitalizing historical photographs, restoring films, and enhancement of the intelligibility of medical images [3].

†Department of Information Engineering, University of Padova,
email: {anil.oezfirat}@studenti.unipd.it
email: {berker.senol}@studenti.unipd.it
email: {ismaildeha.koese}@studenti.unipd.it

In our project, we aim to transform black and white images into color. To achieve this, we utilized deep learning models, specifically Generative Adversarial Networks (GAN) and its variant, Wasserstein-GAN (WGAN). These models learn from the provided images to generate new and unique visuals. GAN consists of two parts: the generator, which produces new images, and the discriminator, which compares the generated images against the real ones to produce a result. Our GANs’s generator is supported by U-net for efficient and accurate image segmentation and uses PatchGAN discriminator as a discriminator to be able to evaluate the input images in more detail [4].

Building upon the work presented in the paper ”Image-to-Image Translation with Conditional Adversarial Networks” [4], we will carry out our study. To enhance the foundational work, we will incorporate the unused WGAN method into a different data-set and observe the results. We will evaluate our results using the FID (Frèchet inception distance), IS (Inception Score), PSNR (Peak Signal-to-Noise Ratio), and SSIM (Structural Similarity) metrics.

II. Related Work

They implement a ”U-Net” based architecture for the generator and a convolutional ”PatchGAN” classifier for the discriminator instead of previous conventional designs. The PatchGAN classifier focuses on the penalizing structure at the level of image patches, as it was initially proposed for capturing local style statistics. In the paper, they demonstrate the effectiveness of this approach across various problem domains and explore the impact of altering the patch size. The findings of this study indicate that conditional adversarial networks show great potential for image-to-image translation tasks, particularly those involving intricate graphical outputs. These networks have the capability to learn task-specific loss functions, making them suitable for diverse applications and datasets [4].

The PatchGAN architecture is determined by the size of its receptive field, also known as the effective receptive field. It focuses on smaller patches of size NxN. In this case, a PatchGAN with a size of 70x70 is employed, which means that each output of the model corresponds to a 70x70 square section of the input image. In essence, a 70x70 PatchGAN classifies patches of the input image measuring 70x70 as either real or fake [4] , [5].

$$\text{receptive field} = (\text{output size} - 1) \times \text{stride} + \text{kernel size}$$

Then apply the discriminator to different parts of the image and combine the results to get the final output of the discriminator [4].

Martin Heusel et al. was proposed "Frechet Inception Distance" (FID score) as a metric to evaluate GANS, which captures the similarity between generated images and real ones in their 2017 paper named "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". The FID score was an improvement over Inception Score, or IS. Lower FID means smaller distances between synthetic and real-world data distributions(two Gaussians). The result of lower FID is better-quality images, and also, a high score indicates lower-quality images and the relationship may be linear [6]. The stability of FID remains intact even when employing a more substantial modification to the encoding, such as using a network trained for a different task. However, it is likely possible to deceive FID by introducing artifacts specifically designed to exploit the encoding network [7].

Ballester and colleagues introduced "Analysis of Different Losses for Deep Learning Image Colorization" for coloring images based on GAN method. They utilized a U-Net architecture as the generator and a Wasserstein GAN (WGAN) as the discriminator, employing various loss functions. The model was trained on images from the COCO dataset, and their results indicated that the VGG-based LPIPS loss function achieved slightly better performance. The study showed that when using Wasserstein GAN combined with the L2 loss or the VGG-based LPIPS loss, the colorized images appeared to have more vivid colors [8].

III. Processing Pipeline

Dataset:

In this part, we will discuss the preprocessing steps performed on the images before training the model. These steps involve ensuring a consistent size for all images, normalizing the pixel values, and utilizing an appropriate color space for the deep learning project. The images were resized to a consistent size of 256x256 and normalized to bring the pixel values to a standard range, enabling the model to learn more effectively while avoiding bias from extreme pixel values. Additionally, as recommended in the Pix2Pix study [4], a widely recognized paper in the field of image-to-image translation, the images were converted to the LAB color space. This conversion allows the model to better process and understand the color information in the images, contributing to improved performance.

U-Net:

The U-Net is an architectural design of a convolutional neural network specifically developed for efficient and accurate image segmentation with limited training data. It adopts a U-shaped structure, comprising an encoder network and a decoder network.

The encoder network captures the features of down-sampled images (NxN) through the utilization of convolutional layers and max pooling. By incorporating multiple convolutional and pooling layers to reduce dimensionality, more intricate details can be extracted. On the other hand, the decoder network employs transpose convolution (deconvolution) and concatenates it with the corresponding skip connection feature map from the encoder block. These skip connections enable the transfer of information from earlier convolutional layers to the deconvolution layers, compensating for the potential loss of features caused by the depth of the network [9].

In our project, we used U-net as a generator. We feed the encoder 256x256 L-channel (grayscale) images. With each layer, the reducing spatial resolution via downsampling by a factor of 2. This process continues until the image reaches a dimension of 1x1. At the decoder part, the decoder blocks of the network upsample the image to have the same output size as the input. The final output of the decoding process consists of the generated LAB-channel version of the initial input. An example U-Net model Architecture can be seen in Fig 1.

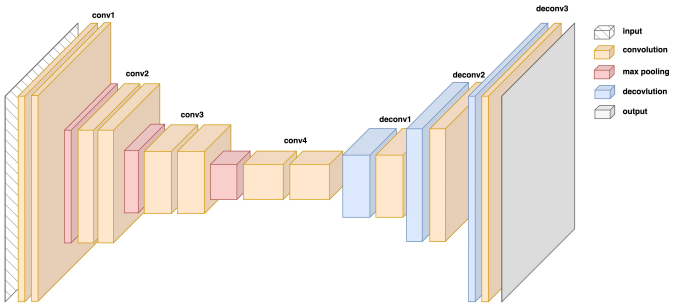

Fig. 1: U-Net Architecture

PatchGan Discriminator:

PatchGAN discriminator is a type of GAN discriminator that classify the images patch by patch as real or fake. The difference from the traditional discriminator is actually output. In traditionally discriminator's CNN, gives the probabilities of a whole input image, but it is different from PatchGAN. It gives probabilities of it is real or fake patches of the input image as independently for each patch and it is used NxN outputs vector. The overall result was made by calculating the average of the predictions made for all the patches. The values for the patches are between 0 and 1, 0 for fake and 1 for real. By using it, our discriminator only penalizes structure at the scale of image patches. Thus, PatchGAN has fewer parameters, and runs faster [4], [10].

PatchGAN is trained to maximize the adversarial (GAN) loss. We have trained the discriminator well, we used L1 and adversarial (GAN) loss as a first part of our model. Afterward, we have trained the model again by using Wasserstein (WGAN) loss and L1 loss together to achieve better results. An example PatchGAN model Architecture can be seen in Fig 2.

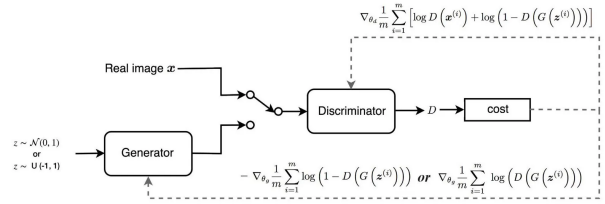

Fig. 2: PatchGan Discriminator Architecture

GAN Loss:

GAN has two models which are a generator and a discriminator model which learn to solve a problem together. Mathematically, consider x as the gray-scale image, z as the input noise for the generator, and y as the 2-channel output that we want from the generator as output. Also, G is the generator model and D stands for the discriminator model. Then the loss function for our GAN becomes [4]:

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

When G tries to minimize this objective against and adversarial D that tries to maximize it, mathematically it can be written as:

$$G^* = argmin_G max_D \mathbb{L}_{cGAN}(G, D)$$

L1 Loss:

While GAN loss focuses on overall image quality, it does not explicitly consider pixel-level differences between the generated and target images. This limitation can result in blurred or distorted outputs. To solve this issue, L1 loss as an additional component has been added to the training objective rather than L2 loss because it has less effect of blurring and producing less gray-ish images.

$$\mathcal{L}_{\text{L1}} = \frac{1}{N} \sum_{i=1}^{N} |G(x_i) - y_i|$$

The combined loss function can be mathematically written as:

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

The coefficient Î controls the contribution of the two losses to the final loss.

WGAN Loss:

WGAN loss function has been introduced by Arjovsky et al. in 2017 to address the limitations of the GAN loss function by introducing a new perspective with the utilization of Wasserstein distance, a measure of dissimilarity between probability distributions for the training process. Which can be mathematically shown as:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma}\left[\, \|x - y\| \,\right]$$

In the context where $\in \hat{a}$ Q, representing the set of all possible joint probability distributions

between Pr and Pg, $\in \hat{I}^3(x, y)$ quantifies the amount of mass required to be transported from x to y in order to transform the distributions Pr to Pg. The WGAN value function is formulated by leveraging the principles of Kantorovich Rubinstein duality. We can achieve the function at the below:

$$\max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z)]$$

The function represented above corresponds to the objective function employed by the discriminator network. In conventional Generative Adversarial Networks (GANs), such as vanilla GAN, the network's objective is to minimize the discrepancy between the distributions of real and generated images in order to learn the desired mapping. To measure this discrepancy, the Vanilla GAN utilizes the KL divergence metric. However, a drawback of this approach is that as the generated data distribution diverges from the real data distribution, the gradient diminishes. Consequently, this leads to vanishing gradients and hinders effective learning for the network.

To overcome this limitation, the Wasserstein Generative Adversarial Network (WGAN) was introduced as a solution for the KL divergence problem encountered in Vanilla GANs. Notably, by employing the Wasserstein distance, WGANs address the limitations of GANs, such as vanishing gradients and mode collapse. It provides more stable and reliable gradients, granting more effective learning for both the generator and discriminator networks. According to the literature, WGANs are proposed to provide more stable gradients compared to Vanilla GANs and consequently leading to improved results.

Quality Metrics:

FID is a metric to evaluate GANS, which captures the similarity between generated images and real ones and FID score is an improved metric compared to the Inception Score (IS) for evaluating image quality. A lower FID score indicates that the synthetic and real-world data distributions are closer together, resulting in higher-quality images [6].

$$\text{FID} = \|\mu_{\text{real}} - \mu_{\text{fake}}\|_2^2 + \text{Tr}(\Sigma_{\text{real}} + \Sigma_{\text{fake}} - 2(\Sigma_{\text{real}}\Sigma_{\text{fake}})^{1/2})$$

Peak Signal-to-Noise Ratio (PSNR) is a measure of the ratio between the reference image pixels and the noise introduced during processing. In this experiment, PSNR is used to assess how well processing and correction methods restore a degraded image to its original state. The reference images are assumed to be perfect with no noise, resulting

in an infinite PSNR score. The processed or reconstructed images are compared to their respective baseline images. A high PSNR indicates an effectively reconstructed image, while a low score suggests the persistence of distortion after reconstruction [11]. For the following implementation, let us assume a standard 2D array or matrix of data. It is important to note that the dimensions of both the correct image matrix and the degraded image matrix should be the same. The mathematical representation of the Peak Signal-to-Noise Ratio (PSNR) is provided below.

$$\text{PSNR} = 20 \cdot \log_{10}(\text{MAX}) - 10 \cdot \log_{10}(\text{MSE})$$

where the MSE (Mean Squared Error) is:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

MSE and subsequent PSNR cannot be computed when comparing two identical images as it results in a division by zero. This metric is limited because it only focuses on numerical comparison and fails to consider biological factors of the human vision system, such as the Structural Similarity Index (SSIM), which accounts for structural similarities.

SSIM is a metric that measures the statistical similarity between two images. The processed images are then compared to this reference using the SSIM metric. We use the mean SSIM (MSSIM) index, as described by Wang, Bovik, Sheikh, and Simoncelli, to evaluate the overall image quality, considering that the local application of SSIM provides a more accurate representation of statistical features in an image. By using the SSIM index, we aim to quantify the similarity between images before and after successive rounds of processing. The expected behavior of this metric is a gradually decreasing index value, which is consistent with the amount of applied processing [11].

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

IV. Signals and Features

In this project, we utilized two datasets, namely ImageNet-1000 and Coco2017 Dataset. Initially, we downloaded and added the datasets to Google Drive for convenience. Moreover, our model was not carried on Google Colab Pro as we expect. The pull request by Google Drive was disabled due to unexpected problems and we decided to initiate the model on the local disk. Each dataset consists of images with diverse contexts and resolutions. To ensure consistency, our model resized all the images to a fixed size of 256x256 pixels using bicubic interpolation. Additionally, we converted the images to the LAB colorspace and normalized the values between -1 and 1 to achieve better results. The decision to convert the images to LAB was based on its ability to yield improved predictions. The LAB colorspace distinguishes between three channels: L, a*, and b*. The L channel relates to luminance, representing the grayscale version of image, while the a* and b* channels encode the main colors such as green-red and blue-yellow, respectively. By leveraging LAB instead of RGB, we benefited from perceptual uniformity, separation of luminance and chromaticity, robustness to lighting conditions, reduced dimensionality, and color-based image analysis.

To further proceed, we partitioned the datasets into two sets: one for training purposes and the other for testing. The distribution of images for the training process accounted for approximately 85-90% of the dataset, while the remaining 10-15% was allocated for testing.

V. Learning Framework

In this section, we represented the learning strategies and the algorithms employed to tackle for the task of image colorization. We have used Generative Adversarial Networks (GANs) and specifically the Wasserstein GAN(WGAN) approach to compare two different models. Specifically, our approach involved a generator network and it is based on the U-Net architecture. U-Net architecture is a fully convolutional network and got its name from its U-Shaped design, which consists of an encoder and decoder layers. The encoder part of the architecture performs downsampling on the input image. By doing that process on the input data, it reduces the spatial dimensions of the input feature map while retaining the essential information. The encoder layers handle the upsampling of the feature maps. Which is the inverse of the encoder part by increasing the spatial dimensions of the feature map while preserving important information and maintaining spatial relationships. The generator which has been built with U-Net architecture takes a gray-scale image as input and produces a color image of the same resolution. In our implementation of the article, our U-Net consisted of 8 down-sapling and up-sampling layers, with the initial convolutional layer having 64 filters. Notably, skip connections were incorporated between corresponding encoder and decoder layers to improve the information flow and gradient propagation within the network, leading to better performance and training stability. For the generator part of the architecture, we designed it by using the PatchGAN architecture. Which aims It aims to classify local patches of an image as real or fake, instead of providing a single scalar output unlike traditional GANs that output scalar values (0 or 1) to classify images as real or fake, the PatchGAN performs a fully convolutional evaluation on the entire image and outputs an specified NxN matrix, representing averaged responses over patches. Our implementation of the PatchedGAN architecture compromised with 3 downsampling layers with 64 filters. Both the generator and discriminator networks were initialized with random weights and trained on the training set. During the training process, the performance of the generator network was evaluated using the validation set. The PatchDiscriminator and U-Net scheme presented in this project enabled effective image colorization by leveraging the strengths of U-Net's encoder-decoder structure and PatchGAN's patch-level classification capabilities.

Pix2Pix:

The Pix2Pix GAN model, introduced by Isole et al. [4], is an image-to-image translation algorithm that generates realistic color images from grayscale input images. In the Pix2Pix architecture, a U-Net generator is implemented. The U-Net takes the grayscale image as input and generates corresponding colored images. On the other hand, PatchDiscriminator is used as the discriminator network, which aims to distinguish between real and fake images. The generator minimizes a combination of two-loss functions: adversarial loss and L1 loss. The adversarial loss is calculated using the GANLoss criterion, which encourages the generated images to be indistinguishable from real images. The L1 loss, calculated using the L1Loss criterion, measures the pixel-wise difference between the generated image and ground the truth image. The discriminator network is trained to maximize the adversarial loss, aiming to correctly classify real and fake images. The pipeline of Pix2Pix GAN, along with its loss function, can be visualized in Figure 3 of the paper.

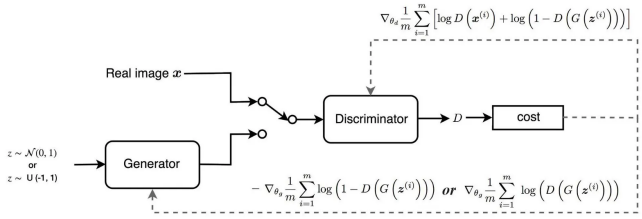

Fig. 3: Generative Adversarial Network (GAN) Architecture

Wasserstein GAN (WGAN):

In the WGAN model, we utilized the U-Net architecture as the generator and utilized the PatchDiscriminator as the discriminator, similar to the Pix2Pix GAN model. The key difference between WGAN and GAN is loss function which using on generator and discriminator. Specifically, the WGAN model utilizes the Wasserstein distance (Earth Mover's distance) as a metric for quantifying the between two probability distributions in a given metric space. This distance can be interpreted as the minimum amount of effort required to transform one distribution into another, where effort is defined as the product of the mass of the distribution that needs to be moved and the distance it needs to be moved. By employing the Wasserstein distance as the loss function, the WGAN model achieves more stable training for both the generator and the discriminator. Additionally, it has a property that the gradient of the loss function is always well-defined, enabling more stable optimization [12].

In the image recoloring project, the generator is taken grayscale image (L-Channel image) and is still trained to generate realistic color images (combined the generated AB channels and input image as RGB). At the same time the discriminator is trained to approximate the Wasserstein distance between the real and fake images. In this pipeline, the critic refers to our PatchDiscriminator. In accordance with the "Wasserstein GAN" paper [12], the generator and discriminator networks are trained by utilizing the Root Mean Squared propagation (RMSprop) optimizer, employing a learning rate of 5e-5. The RMSprop Optimizer's stability helps prevent oscillations and ensures smoother convergence. By effectively managing the learning rate, RMSprop contributes to stable training of the critic and generator networks, facilitating the optimization process in WGANs [12], [13]. The pipeline of the WGAN with its loss functions can be seen in Fig 4. In this pipeline, the critic refers to our PatchDiscriminator.

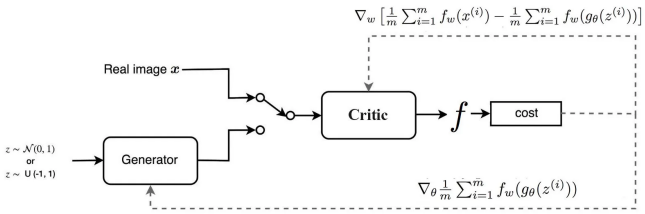

Fig. 4: Wasserstein GAN (WGAN) Architecture

VI. Results

In this study, we compare the performance of image recoloring using Vanilla GAN and WGAN. We also trained the our models with the same parameters but different datasets which are COCO and mini ImageNet-10000 to assess adaptability. Based on FID, PSNR, and SSIM scores, observed our GAN model efficiently adapts to other dataset. However, for WGAN, the results are acceptable for the ImageNet Dataset, but the results are noticeably better for the COCO Dataset.

ImageNet performs two times slower to complete the same number of epochs in training than COCO Data-set, because of that the largence of data-sets. We have trained both models with 100 epochs. For COCO, Vanilla GAN it took about 4 hours while for WGAN it took about 5 hours to train the models, for ImageNet, Vanilla GAN it took about 8 hours while for WGAN it took about 10 hours to train the models. According to the evaluation results presented in Table 1, it is evident that the Vanilla GAN outperforms the WGAN in terms of FID, IS, PSNR, and SSIM scores. A lower FID score indicates a superior outcome, whereas higher IS, PSNR and SSIM scores signify better quality in the generated images.

The outputs of models shown as Figure x and Figure y for COCO Dataset and Figure z and Figure q for ImageNet Dataset. For all outputs figures, first row represent the input (L-Channel image), second row represents model's output (AB-Channel image), the last row represents the ground truth (RGB transformed image of the combined lab image).

Fig. 5: Coco GAN Test

Fig. 6: Coco WGAN Test

Fig. 8: Coco WGAN Train

Fig. 7: Coco GAN Train

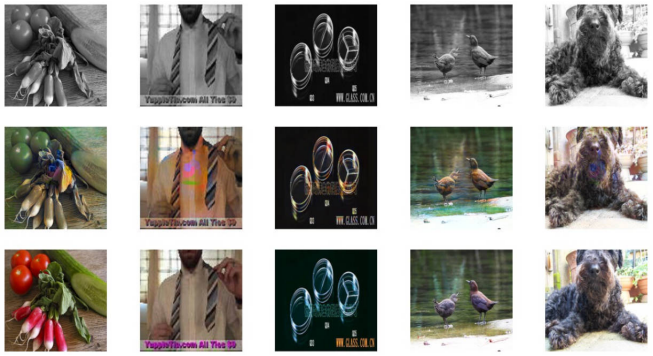

Fig. 9: ImageNet GAN Test

Fig. 10: ImageNet WGAN Test

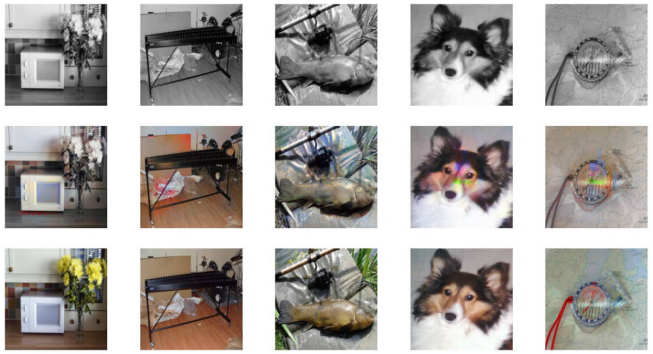

Fig. 11: ImageNet GAN Test

Fig. 12: ImageNet WGAN Test

In terms of visual representation, the COCO Dataset reveals that Figure 5 outperforms Figure 6, particularly when comparing the second image's outcome. The WGAN image displays colors that are significantly different from the real image, appearing blue and making it difficult to distinguish the object from its background. Conversely, the vanilla GAN produces colors that are closer to the real image. Similarly, at the ImageNet Dataset, we observe that Figure 9 is superior to Figure 10, especially when evaluating the second image's result. The WGAN image again exhibits colors that deviate from the real image, appearing green and lacking clear distinction between the object and its background. On the other hand, the vanilla GAN generates colors that closely resemble the real image.

Based on the evaluation of metrics such as FID, PSNR, SSIM, and visual inspection, it can be concluded that the vanilla GAN outperforms the WGAN in the task of recoloring images. The WGAN model did not achieve better results and actually exhibited inferior performance compared to the vanilla GAN for both datasets. The results indicate that the model demonstrates a good level of generalization, as its performance on the test dataset closely matches that on the training dataset. This suggests that the model avoids overfitting to the training data and is capable of performing well on unseen data.

VII. Concluding Remarks

In this project, our aim was to implement different deep learning architectures and datasets to compare to have more insight for image recoloring task. We implemented U-Net the for the generator and a Patch-GAN as the discriminator, During the experimentation stage of the project, we have used different loss functions such as Vanilla GAN, L1, and Wasserstein Generative Adversarial Networks (WGAN). We evaluated the performance of our model using FID, PSNR, SSIM and Inception Scoring on a COCO and ImageNet datasets. Our results showed that the Vanilla GAN architecture outperformed the WGAN architecture in terms of time and quality of the generated images. The FID, PSNR, SSIM and Inception scores were better for the Vanilla GAN model, and the visual analysis also supported these results. In this study, We used the standard WGAN and compared with vanilla GAN with different datasets, it is less likely to have a vanishing gradients problem, it can still have and might be not always optimal. In this case, also WGAN with a gradient penalty can be used. However, it is worth noting that our results should be taken carefully as we had limited resources such as using the free version of Google Colab and a small datasets. So, further research can be done by trying different datasets and training for more epochs. Additionally, As future work, we could explore using other architectures such as DenseNet and also adding gradient penalty coefficients to improve the performance of the model. In conclusion, we have successfully implemented image recoloring using a GAN and WGAN model and evaluated its performance using various quality assessment metrics for different datasets. Through this project, we learned how to implement GAN and had hands-on experience with PyTorch. The challenging part of this work was to improve the performance after implementing the model and applying different parameters. We have worked together for the implementation of the code and to write the report.

Test/Train	Type of GAN	FID	PSNR	SSIM	Inception
Test	Vanilla GAN	0.038	21.31	0.8664	12.66
Test	WGAN	0.086	17.5777	0.7385	12.386
Train	Vanilla GAN	0.026	22.17	0.87	12.034
Train	WGAN	0.039	17.14	0.8264	11.482

[Image Dataset Table 1]

References

[1] Y. Pang, J. Lin, T. Qin, and Z. Chen, "Image-to-image translation: Methods and applications," *IEEE Transactions on Multimedia*, vol. 24, pp. 3859–3881, 2022.

[2] I. Zeger, S. Grgic, J. Vukovic, and G. Sisul, "Grayscale image colorization methods: Overview and evaluation," *IEEE Access*, vol. 9, pp. 113326–113346, 2021.

[3] M. S. Khan, Y. Gotoh, and N. Nida, "Medical image colorization for better visualization and segmentation," pp. 571–580, 07 2017.

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 11 2016.

[5] J. Brownlee, "How to implement pix2pix gan models from scratch with keras," 07 2019.

[6] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 06 2017.

[7] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," *arxiv.org*, 11 2017.

[8] C. Ballester, A. Bugeau, H. Carrillo, M. ClÃ©ment, R. Giraud, L. Raad, and P. Vitoria, "Analysis of different losses for deep learning image colorization," 05 2022.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[10] U. Demir and G. Unal, "Patch-based image inpainting with generative adversarial networks," 03 2018.

[11] C. M. Ward, J. Harguess, B. Crabb, and S. Parameswaran, "Image quality assessment for determining efficacy and limitations of super-resolution convolutional neural network (srcnn)," 09 2017.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[13] N. S. , G. Hinton and K. Swersky, ""neural networks for machine learning lecture 6a overview of mini-batch gradient descent"," 07 2012.