Image Recoloring with conditional GANs

Shayan Sharifi†, Mohammad Vanai†, Shahla Sadeghzadeh†

***Abstract*—Image colorization presents a captivating challenge within the realm of computer vision, entailing the transformation of an image from a source domain to a target domain with the goal of closely mimicking its natural color palette. In the pursuit of colorization, the primary objective is to generate a faithful color image that effectively serves the user's intended purpose, even if it deviates from the ground truth colors. Initially, we start by replicating the methodology originally introduced by Isola et al, which led to pix2pix framework. They have offered GANs in a conditional setting as a general-purpose solution to image-to-image translations. To enhance our results, we first experimented with ResNet and retrained the network. Subsequently, we implemented WGAN-clipping and WGAN-PG to enhance the stability of the learning model. Additionally, we applied CycleGAN to improve the speed of computational processing. Subsequently, we evaluated these three image colorization models using PSNR and SSIM metrics. Our contribution seeks to advance the image recolorization task by incorporating training enhancements that significantly reduce the requisite dataset size, thereby enhancing both speed and the logical coherence of colorization outcomes.**

***Index Terms*—Supervised Learning, Generative Adversarial Network, CGAN, WGAN-clipping, WGAN-pg, CycleGAN, PatchGAN, and U-Net.**

I. Introduction

graphicx

Image colorization, which involves the process of adding colors to grayscale images, has recently attracted considerable attention in the field of computer vision. This is due to its wide range of applications, including color restoration, automatic animation coloring, medical images enhancement, and improvements in cartography. During the process of colorization, grayscale images are transformed into colorful three-dimensional images. Image colorization research generally falls into two main categories: user-guided and automatic methods. In the user-guided category, color information is provided by users or reference images to guide the coloring process [1]. In the second approach, Convolutional Neural Networks (CNNs) have been extensively explored and employed for the automatic colorization task. Since CNNs have good feature extraction and high differentiation capability, therefore it is used primarily in feature extraction/generation stages in pattern recognition systems [1]. In contrast, the limitations of Convolutional Neural Networks (CNNs), such as their difficulty in handling color context, uncertainty, reliance on ground truth data, potential loss of fine details, and challenges in achieving realism, motivate the use of Generative Adversarial Networks (GANs) for image colorization

Generative Adversarial Network (GAN) is a generative model based on game theory [2] which is consist of two smaller networks: the generator and the discriminator. The generator's objective is to create images that are so realistic that the discriminator cannot distinguish them from real ones [3]. In relation to Figure 1.

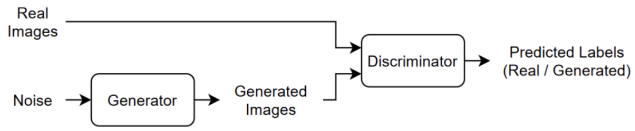

Fig. 1: Generative Adversarial Network (GAN) structure diagram

In this article, we employ conditional Generative Adversarial Networks (cGANs), which excel at converting an input image into a corresponding output image. These networks have showcased impressive performance, particularly when trained with paired data [1]. This model incorporates conditional information during both the training and generation processes. While a standard GAN's generator network typically relies on random noise as input image to produce output image, in the case of a cGAN, the generator goes beyond noise input and also takes additional information into account. This additional information is often presented in the form of labels or class details. cGANs find extensive application in tasks such as image-to-image translation, text-to-image synthesis, artistic style transfer, and more.

We have used the pix2pix [12] model which has been able to provide good results in image-to-image translation and real image generation. Then we will explore an alternative strategy that diverges from the conventional classification and regression paradigms.

The remainder of the paper is structured as follows: In Section II, we introduce the related work. In Section III, we present our pipeline, followed by an explanation of the new generator in Sections IV and V, respectively. Finally, we provide our conclusions in Section VI.

II. Related Work

Image-to-image translation is a field where numerous researchers have harnessed adversarial learning. The primary objective is to transform an input image from one domain into another domain, and this is achieved through the use of training data containing pairs of input and output images. In comparison to the L1 loss, which often results in the production of blurry images [4], the adversarial loss has gained significant popularity across various image-to-image tasks.

The rationale behind this preference lies in the fact that the discriminator, in the adversarial learning framework, can effectively learn a dynamic and adaptable loss function. This

†Department of Information Engineering, University of Padova, email: {shayan.sharifi}@unipd.it

loss function automatically adjusts to the discrepancies between the images generated by the model and real images from the target domain. A notable example of this approach is the recent pix2pix framework, which utilizes image-conditional Generative Adversarial Networks (GANs) [5]. A cGAN is a generative model consisting of a vanilla GAN framework along with additional external control factors, such as class labels, reference images, object keypoints, human skeletons, and semantic maps [2].

In this paper, our main focus is around the recoloring task using cGAN. In general, contemporary image-to-image translation methods can be classified into two main types: supervised/paired and unsupervised/paired [2]. While the basic model shows a primary understanding of common objects in images, such as the sky and trees, the output it produces is not very visually pleasing. It has problems with determining the color for rare objects and has issues such as color spillovers and circular color clumps. Consequently, it appears that it is difficult to achieve satisfactory results with this strategy, especially with limited data sets. Thus, we have implemented a new approach in cGANs Generator by using ResNet, which is elaborated in the Learning Framework section of Part V. Furthermore, we have implemented two different structure of Wasserstein GANs: WGAN-Clipping, which involves the use of weight clipping, and WGAN-GP, which incorporates Gradient Penalty. Both are employed to enhance training stability [6]. There is a common architecture for the generator (U-Net structure) and discriminator (Patch-GAN) with minor differences in specific components, which will be explained in future sections. The final step involves another implementation of GAN structure to achive higher processing speed.

III. Processing Pipeline

Our image recovery pipeline, using a Conditional Generative Adversarial Network (CGAN), consists of four distinct stages, as illustrated in 2: data acquisition, data preprocessing, model implementation, and result analysis and decision-making.

In the first step, we use an existing dataset known as COCO, which contains a staggering collection of over 300,000 images, each labeled with one of 80 unique object categories or classes. From this vast dataset, we select a subset of 10,000 images to serve as the basis for our training process.

The second step includes preprocessing the selected images. Here, we resize the images and apply a horizontal flip if they belong to the training set. Subsequently, we convert these images into both grayscale and Lab color spaces. After this transformation, we perform normalization and separate the resulting grayscale channel and color channels. These channels serve as the inputs and targets, respectively, for our learning model.

The third step is dedicated to the actual training of our CGAN model. This model consists of two integral components: the generator and the discriminator. The main goal of this model is to produce images that closely resemble the originals.

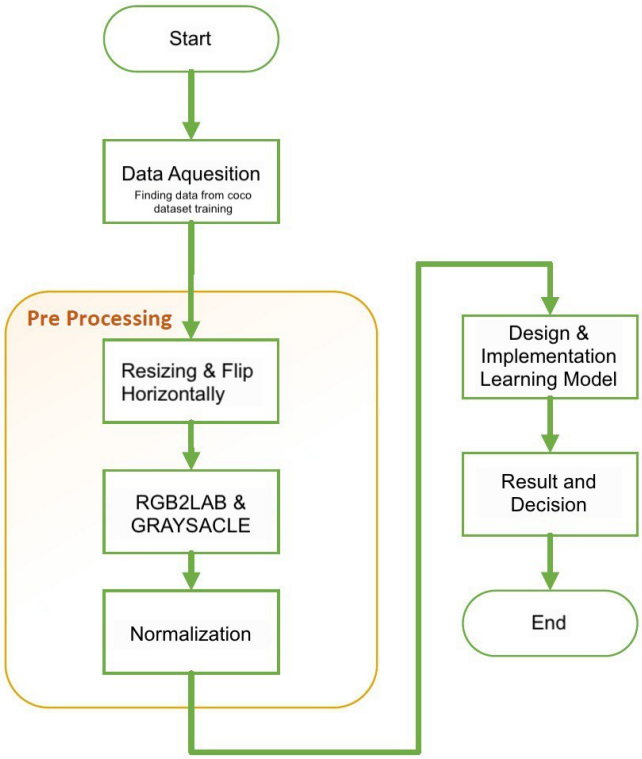

Fig. 2: Processing Pipline

Finally, the fourth step involves result analysis and decision-making. Here, we evaluate the outcomes of our image recovery process and make informed decisions regarding any necessary improvements or adjustments.

IV. Signals and Features

In this project, we used a subset of the COCO dataset, consisting of 10,000 images. In this subset, 8000 images were allocated for training and the remaining 2000 images for testing. Beyond the inherent diversity and comprehensiveness of the dataset, COCO is recognized as an established resource in the computer vision community. Its widespread use makes it a valuable asset for comparison with other research and models.

During the initial phase of our project, we preprocessed the selected data. All images were resized to standard dimensions of 256x256 pixels and subsequently converted into tensors (note that all training images were flipped horizontally at this stage). To achieve our goal of recoloring the images, we further converted them from the RGB color space to the LAB color space domain. ”To normalize the pixel values to a standardized range, we applied a normalization process and set the values in the range of -1 to 1 in the LAB color space.”

V. Learning Framework

In the ’Image-to-Image Translation with Conditional Adversarial Networks’ paper, also known as ’pix2pix’, a comprehensive solution is proposed for various image-to-image tasks within the realm of deep learning. Notably, one of the tasks addressed by this approach is colorization [5].

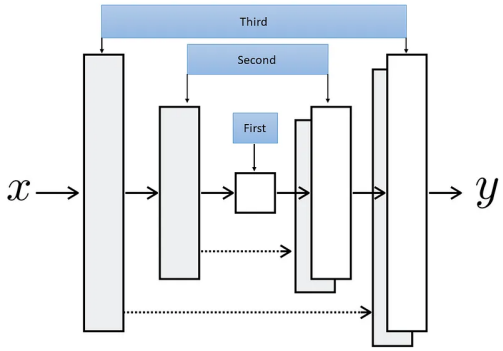

Fig. 3: U-Net Structure

The pix2pix framework employs two essential loss functions to achieve its objectives. Firstly, the L1 loss, which frames the problem as a regression task, aims to minimize the difference between the predicted outputs and the ground truth. Secondly, the adversarial loss, inspired by Generative Adversarial Networks (GANs), operates in an unsupervised manner. It assigns a numerical value to the quality of the generated outputs, effectively measuring how 'real' they appear.

A. Network Model

We have successfully implemented our initial learning model using Conditional Generative Adversarial Networks (cGAN). As previously highlighted, cGAN models comprise two fundamental components: the Discriminator and the Generator. This architecture serves as a foundational framework for our second model, the Wasserstein Generative Adversarial Network (WGAN). Let's now delve into the specific details and characteristics of each of these components.

U-Net Generator: The Generator is specifically configured with an equal number of downsampling (pooling) and upsampling layers. Moreover, it incorporates skip connections between mirrored downsampling and upsampling layers, similar to forming a U-shaped network [7]. The input image is gradually reduced until it becomes a vector with dimensions of 2x2 pixels. This compressed representation is then expanded through upsampling to match the original input size of 32x32 pixels. This approach is inspired by encoder-decoder networks. This method is useful for training neural networks with limited memory resources and facilitates the sharing of low-level information between input and output [8]. 3 shows the simple structure of Unet.

U-Net with Attention Mechanism in Generative Models

Generative models for image processing have evolved significantly with the integration of attention mechanisms. This method explores the integration of attention into a colorization model, focusing on key challenges addressed and resulting improvements. [9]

In neural networks, attention mechanisms dynamically focus on specific input regions, facilitating nuanced understanding and context-aware computation. [10]

The generator architecture is augmented by seamlessly integrating an attention mechanism. This involves introducing an Attention-Block and incorporating it into the existing Unet architecture, resulting in an enhanced version denoted as Unet-With-Attentionccccc. Grayscale images often contain overlooked details, prompting the need for a dynamic approach. The attention mechanism resolves this by prioritizing specific regions, ensuring a focus on crucial details. Diverse grayscale images pose challenges for uniform colorization models. Attention mechanisms adapt focus based on input characteristics, enhancing the model's ability to handle varied content. Traditional architectures struggle with capturing long-range dependencies, addressed by attention mechanisms that facilitate consideration of relationships between distant pixels [11]. Moreover, realistic colorizations require attention to fine details and plausible color choices, achieved by selectively attending to relevant regions.

Improving Model Robustness: Models without attention mechanisms may struggle to generalize across diverse images. Attention mechanisms enhance model robustness by adapting processing based on input characteristics.

Benefit in Colorization Model: Integrated into the Unet architecture, the attention mechanism enriches the colorization model's ability to adaptively attend to critical regions within grayscale images, capturing intricate details and producing realistic colorizations.

PatchGAN Discriminator: The goal of the discriminator is to differentiate between real samples from the dataset and fake samples generated by the generator. The architecture of the discriminator is relatively straightforward. follows the standard blocks of Conv-BatchNorm-LeakyReLU to determine whether the input image is real or fake. It's important to note that the first and last blocks do not employ normalization, and the last block lacks an activation function (as it is embedded in the loss function we will utilize).

In this instance, we have employed a 'Patch' Discriminator. In a standard discriminator, the model produces a single number (a scalar) that signifies its confidence in the entire input image being real or fake.In contrast, a patch discriminator assesses the input image in smaller patches, typically 70 by 70 pixels each[Unified Generative Adversarial Networks for Controllable Image-to-Image Translation], and makes individual determinations for each patch's authenticity. Utilizing such a model for the task of colorization proves logical, as local changes are pivotal in this context. Deciding on the entire image, as a vanilla discriminator does, might not adequately address the subtleties required for this task.It's worth noting that the model's output shape is 30 by 30, but this does not dictate the patch size. The actual patch size is determined when calculating the receptive field for each of these 900 (30 multiplied by 30) output values, which, in our case, results in a patch size of 70 by 70 pixels.

Training Neural Network: For the implementation of our model, we used the free and open-source Python libraries Tensorflow and fastai. We used the free "Google Colab

GPU" to train the model. We initialized the weights of our model with a mean of 0.0 and standard deviation of 0.02 which are the proposed hyperparameters in the article. we commence training the discriminator using the 'backward-D' method. Here, we supply the discriminator with the fake images generated by the generator. It's essential to detach them from the generator's graph, treating them as constants to the discriminator, similar to real images. We label these fake images as such. Then, we present a batch of real images from the training set to the discriminator and label them as real. The losses for fake and real images are summed, and their average is calculated. Subsequently, we invoke the 'backward' method on the final loss to update the discriminator. Following the discriminator training, we proceed to train the generator. Within the 'backward-G' method, we provide the discriminator with the fake images and aim to deceive it by assigning real labels to these images, thereby calculating the adversarial loss.

Objective Function: The objective function of cGAN can be explained as this equation:

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}x, z[\log(1 - D(x, G(x, z)))] \quad (1)$$

[5]

Parameters: - x: The grayscale input image.
- z: The input noise for the generator.
- y: The two-channel output we seek from the generator, which can also represent the two color channels of a real image.
- G: The generator model.
- D: The discriminator model.
Notice that 'x' is given to both models, which serves as the condition introduced to both players of this game.

B. First Model: CGAN

In the initial implementation of the complete framework, we designed a discriminator and generator based on the approaches outlined in reference [5]. As mentioned earlier, our model consists of two components with different optimization problems: minimizing for the generator and maximizing for the discriminator. Additionally, as mentioned earlier, we incorporate the L1 loss and measure the difference between the predicted two channels and the target two channels, multiplying this loss by a coefficient, typically 100 in our case, to balance the two losses. We then add this loss to the adversarial loss and invoke the 'backward' method of the loss to update the generator."

Loss function: The previously mentioned loss function primarily helps in producing visually appealing, realistic, and colorful images. However, to enhance the model's performance and introduce a level of supervision in our task, we supplement this loss function with L1 Loss, also known as the mean absolute error. This quantifies the disparity between the predicted colors and the ground truth colors.

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

If we only use the L1 loss, the model can still learn to colorize images; however, it tends to be more conservative. Frequently, it defaults to colors like 'gray' or 'brown' when uncertain about the optimal color choice. This behavior arises because, in instances of uncertainty, the model leans towards selecting the average color, aiming to minimize the L1 loss as much as possible. This behavior is somewhat akin to the blurring effect observed when using L1 or L2 loss in super-resolution tasks.

Moreover, we prefer the L1 Loss over the L2 loss (or mean squared error) because it mitigates the tendency to produce grayish images. Therefore, our combined loss function incorporates these considerations:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (3)$$

Here, λ represents a coefficient used to balance the contributions of the two losses to the final loss function. It's important to note that the discriminator loss, as mentioned, does not include the L1 loss in its formulation.

Optimizers: Optimizers are essential for training neural networks as they dictate how the model learns. In this model, we used a popular optimizer called Adam for both the generator and discriminator.

Hyperparameters: The hyperparameters for this implementation are as follows:
- Training data size: 2400
- Testing data size: 600
- Batch size: 20
- Epoch: 10
- λ (regularization parameter): 100
- Learning rate for Generator (lr-G): 2e-4
- Learning rate for Discriminator (lr-D): 2e-4

Opinion: The results show that with this small data set, it is challenging to achieve satisfactory results using the current strategy. To address this issue, we decided to pre-train the generator separately, using a supervised and deterministic approach to circumvent the aforementioned challenges.

CGAN with ResNet: Moreover, we pre-trained ResNet18 [12], which was chosen as the backbone of the U-Net architecture. The "DynamicUnet" module uses this modified backbone to create a U-Net architecture tailored to our specific needs, producing a dual-channel output designed to fit input images of size 256. This was implemented using the "Dynamic U-Net" module from the fastai library.

C. Second Model: WGAN

In this stage, we implemented WGAN (Wasserstein Generative Adversarial Network), a variant of GAN that utilizes the Wasserstein distance as the loss function for both the generator and the discriminator (referred to as the critic). In WGAN, the primary focus is on training the critic to approximate the Wasserstein distance between real and fake data distributions. The generator's goal is to generate samples that result in low

values from the critic. Notably, WGAN ensures that the critic network behaves as a 1-Lipschitz function, indicating that the critic's output should undergo at most a unit change for any small modification in the input.[Martin]

Loss function: We implemented WGAN with two variations in its loss calculation, incorporating methods based on both clipping weights and gradient penalty. The equations for these variations are as follows [6].
1.WGAN-PG :
The objective function for WGAN-PG includes the Wasserstein distance and a penalty term for the gradient norm.

$$\begin{aligned}\mathcal{L}_{WGAN\_PG} =&\mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[D(\hat{x})] - \mathbb{E}_{x\sim P_r}[D(x)] \\ &+ \lambda \mathbb{E}_{\tilde{x}\sim P_{\tilde{x}}}\left[(\|\nabla_{\tilde{x}} D(\tilde{x})\|_2 - 1)^2\right]\end{aligned} \tag{4}$$

Here
- $D(\hat{x})$ is the discriminator output for generated samples
- $D(x)$ is the discriminator output for real samples
- λ is a hyperparameter controlling the strength of the gradient penalty
- $\nabla_{\tilde{x}} D(\tilde{x})$ is the gradient of the discriminator output with respect to interpolated samples $\tilde{x}$.

2.WGAN-Clipping: WGAN-Clipping enforces Lipschitz continuity by clipping the weights of the discriminator. [6]

$$\mathcal{L}_{WGAN\_Clip} = -\mathbb{E}_{x\sim P_r}[D(x)] + \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[D(\hat{x})] \tag{5}$$

Optimizers: In this scenario, the RMSprop and Adam optimizers are utilized in the model of the WGAN-Clipping and WGAN-GP, respectively. .

Hyperparameters: The hyperparameters for this implementation are as follows:
- Training data size: 2400
- Testing data size: 600
- Batch size: 32
- Epoch: 20
- Weight-CLIP = 0.01 - λ (regularization parameter): 100
- Learning rate for Generator (lr-G): 5e-5
- Learning rate for Discriminator (lr-D): 5e-5

Opinion: As a result, WGAN represents a more stable and clear improvement over GANs, but it seems that the computational runtime is longer. .

D. Third Model: CycleGAN

In the third model, we implemented CycleGAN to improve processing speed, distinguishing it from conventional image translation models that require paired training examples. CycleGAN, featuring two generators and discriminators (each for a specific domain), focuses on translating images between domains. Generators aim to perform domain translation, while discriminators distinguish real images from translated/generated ones. A notable feature is CycleGAN's enforcement of cycle-consistency, ensuring that an image from domain A, translated to domain B and then back to domain A, closely matches the original. This cycle-consistency acts as a training regularization term.

Loss function: The overall loss function employed in this method is derived by combining the individual terms of the loss functions with appropriate weights, as given below [13]:

$$\begin{aligned}\mathcal{L}_{\text{total}} =&\mathcal{L}_{\text{GAN}}(G_{\text{gray}}, D_{\text{color}}, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(G_{\text{color}}, D_{\text{gray}}, X, Y) \\ &+ \lambda \mathcal{L}_{\text{cycle}}(G_{\text{gray}}, G_{\text{color}}, X, Y) \\ &+ 0.51\lambda \mathcal{L}_{\text{detail}}(G_{\text{gray}}, G_{\text{color}}, K, Y)\end{aligned} \tag{6}$$

Here, G-gray and G-color denote two generators. G-color takes a grayscale image X as input and produces a colored image as output, while G-gray takes an image Y as input and generates a grayscale image. The logarithm operation is denoted by log(). L-cycle and L-detail represent the losses associated with cycle-consistency and image detail preservation, respectively. It should be noted that in this implementation, we set λ to 0.

Optimizers: In this model, the Adam optimizer is employed with B1 and B2 parameters set to 0.5 and 0.999, respectively. .

Hyperparameters: The hyperparameters for this implementation are as follows:
- Training data size: 2400
- Testing data size: 600
- Batch size: 1
- Epoch: 20
- λ Identity: 0
- λ Cycle: 10
- Learning rate: 1e-5

Opinion: Personally, we experience higher speed in computational performance with this set of parameters. .

VI. Results

A. Metrics for Evaluating Image Colorization Methods

Evaluating different image colorization methods objectively is tricky, but in computer vision, we commonly use two metrics for this Assessment: PSNR (peak signal-to-noise ratio) and SSIM (structural similarity index measure).All in all higher PSNR or SSIM score means the image reconstruction is better[sensor].

PSNR: PSNR is measured in decibels (dB). A higher PSNR values indicate better image quality which means the reconstructed image is closer to the original, with lower distortion or noise.

SSIM: SSIM is a metric that considers how similar two images are by looking at their brightness, contrast, and structure. Unlike MSE and PSNR, it better aligns with how humans perceive similarity. SSIM values range from -1 to 1, with 1 meaning perfect similarity. In simpler terms, a higher SSIM

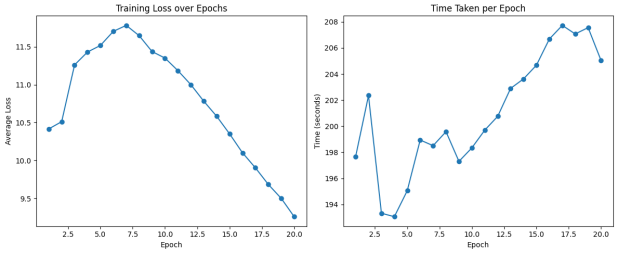

Fig. 4: The output loss time consumption per epoch for CGAN+ Unet-With-Attention for 20 epoch with 3000 data

value indicates that the original and reconstructed images share a closer structural resemblance.

TABLE 1: Evaluation of two metric for two models on the COCO dataset.

	CGAN+ResNet18			Attention-CGAN		
Metrics	AVG	Max	Min	AVG	Max	Min
PSNR	27.70	30.51	23.47	24.67	27.83	16.01
SSIM	0.53	0.70	49/7	0.56	0.56	-0.01

B. Losses and speed comparison

We can make some general observations based on charts 4 and 5 in term of Stability and Convergence Speed: Wasserstein GANs (WGAN-CP, WGAN-GP) are designed to address training stability issues associated with traditional GANs, potentially leading to faster convergence. Residual networks (ResNet) are known for facilitating the training of deep networks, which might accelerate convergence. Cycling GAN: Cycling GANs involve two GANs cycled in a coupled manner, which may introduce additional complexity. The training speed could be affected by the need for multiple generators and discriminators. Conditional GANs (CGAN): The inclusion of conditional information in CGANs might provide better guidance for the generator, potentially aiding convergence.

Here, we can observe some output images from the primary U-Net in Figure 6, and U-Net with ResNet18 for 10

TABLE 2: Evaluation of two WGAN models on the COCO dataset

	WGAN˙GP			WGAN˙Clipping		
Metrics	AVG	Max	Min	AVG	Max	Min
PSNR	18.79	24.29	16.50	21.56	26.08	16.66
SSIM	0.01	0.09	-0.01	0.07	0.20	-0.00

TABLE 3: Evaluation of two CycleGAN models on the COCO dataset

	CycleGAN		
Metrics	AVG	Max	Min
PSNR	6.03	11.12	1.69
SSIM	-0.01	0.41	-0.36

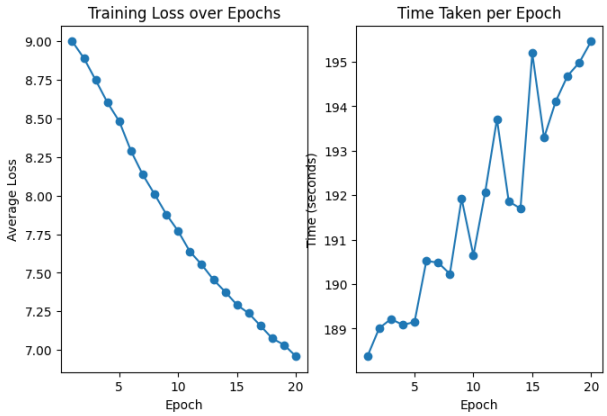

Fig. 5: The output loss time consumption per epoch for CGAN+ResNet18- Attention for 20 epoch with 3000 data

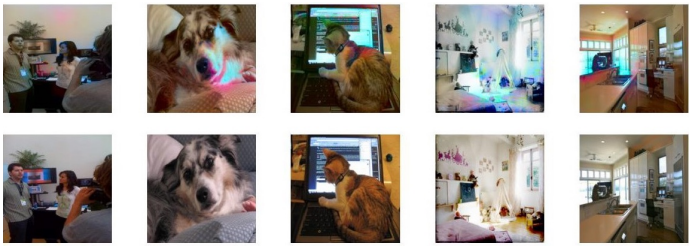

Fig. 6: First row: the output images from the primary U-Net, Second row: Ground truth images

epochs with 5000 and 100000 in Figure **??** and 7respectively.

VII. Concluding Remarks

In conclusion, our study addresses the captivating challenge of image colorization by advancing the methodology inspired by Isola et al., initially introduced in the pix2pix framework. We experimented with ResNet, implemented WGAN-clipping, WGAN-PG, and applied CycleGAN to achieve stability, computational speed improvement, and model robustness. Evaluation using PSNR and SSIM metrics revealed notable enhancements in the recolorization task.

Contrary to the limitations encountered with the pretrained U-Net architecture in our previous findings, our updated methodology incorporates U-Net with an Attention Mechanism in Generative Models. This integration significantly

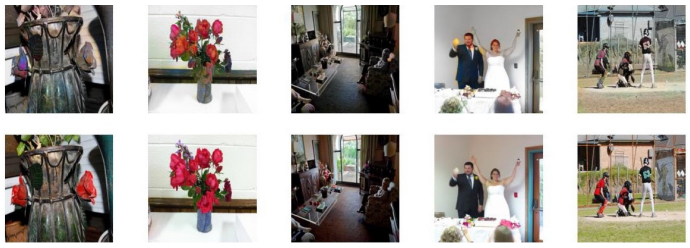

Fig. 7: First row: the output images from U-Net with ResNet18 for 10epoch with 10000 data, Second row: Ground truth images

Fig. 8: First row: the output images from U-Net with ResNet18 for 20 epoch with 3000 data, Second row: Ground truth images

Fig. 9: First row: the output images from CGAN-Resnet weights for 20epoch with 3000 data

improves object and region recognition, leading to superior colorization accuracy and processing speed, even with smaller datasets.

Our contribution extends beyond prior work, reducing the requisite dataset size while enhancing both speed and logical coherence in colorization outcomes. These advancements provide valuable insights for researchers aiming to improve image recolorization approaches and elevate the overall user experience. Further research avenues may explore additional enhancements to boost model performance in diverse scenarios.

Fig. 10: First row: the output images from CGAN-Unet with attention for 20epoch with 3000 data

Fig. 11: First row: the output images from WGAN-Gradient Penalty for 20epoch with 3000 data

Fig. 12: First row: the output images from WGAN-Clipping weights for 20epoch with 3000 data

VIII. CHALLENGES THROUGH WORK

During this project, we faced challenges related to hardware limitations and computing power. We had contemplated the possibility of testing our model with a larger dataset to evaluate its performance. However, due to time and computational constraints, we could not pursue this approach.

Future work could focus on scaling these models to effectively process high-resolution images while preserving complex details and textures. In addition, the development of models optimized for historical photographs or medical images has the potential to provide more accurate and relevant colorization results.

REFERENCES

[1] L. Kiani, M. Saeed, and H. Nezamabadi-pour, "mage Colorization Using Generative Adversarial Networks and Transfer Learning," in *2020 International Conference on Machine Vision and Image Processing (MVIP)*, (Tehran, Iran), Feb. 2020.

[2] H. Tang, H. Liu, Member, and N. Sebe, "Unified Generative Adversarial Networks for Controllable Image-to-Image Translation," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, pp. 8916–8917, 2020.

[3] T. Park, M.-Y. Liu, T.-C. Wang, and J. yun Zhu, "semantic Image Center size with a specialty-adaptive normalization," *Computer Vision Foundation*, pp. 2337–2338, 2019.

[4] Ting-Chun, W. Ming-Yu, L. Jun-Yan, Z. A. Tao1, J. Kautz1, and B. Catanzaro, "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs," *Computer Vision Foundation*, pp. 8799–8800, 2017.

[5] P. Isola, J.-Y. Zhu, T. Z. Alexei, and A. Efros, "Image-to-image translation with conditional adversarial networks," *in Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[6] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, "IMPROVING THE IMPROVED TRAINING OF WASSERSTEIN GANS: A CONSISTENCY TERM AND ITS DUAL EFFECT," *Published as a conference paper at ICLR*, 2018.

[7] M. G. Zili Yi1Ùˆ Hao Zhang, Ping Tan, "Unsupervised dual learning for image-to-image translation," *IEEE International Conference on Computer Vision*, pp. 2868–2869, 2017.

[8] P. R. S. K. MAeshita Mathur, Ameesha Dabas, "Recoloring Grayscale Images using GAN," 2023.

[9] Oktay, "Attention U-Net Learning Where to Look for the Pancreas," 2018.

[10] Jaderberg, "Spatial Transformer Networks," 2015.

[11] Chaudhari, "Attention Mechanism in Neural Networksâ€"Insights and Applications," 2020.

[12] S. R. J. S. Kaiming He, Xiangyu Zhang, "Deep residual learning for image recognition," *Computer Vision Foundation*, pp. 774–775, 2015.

[13] S. Huang, X. Jin, Q. Jiang, J. Li, , S.-J. Lee, P. Wang, and S. Yao1, "A fully-automatic image colorization scheme using improved CycleGAN with skip connections," *Multimedia Tools and ApplicationsR*, 2021.