Image Colorization with Conditional WGAN

Maryam Gholami Shiri†, Amir Mahdi Amani†,Hamid Mohammadi†

***Abstract*—In our research, we explore the application of Conditional Wasserstein Generative Adversarial Networks (cWGANs) for the task of image colorization. Our goal is to transform grayscale images into their high-dimensional, color counterparts. We aim to enhance the Pix2Pix model by incorporating a Wasserstein GAN (WGAN) instead of the traditional GAN architecture. Additionally, we utilize a U-Net architecture based on residual blocks to further improve Pix2Pix's performance. Our proposed model accepts grayscale images as input and generates corresponding colorized images. We trained and evaluated our models using the MIRFLICKR25k dataset, observing how increasing the number of epochs directly influenced Generator and Critic losses. To regularize the activations and prevent overfitting, we employed Dropout2d in ResBlock and in the encoding process modules. Our research contributes to the field of image colorization by providing an improved model and methodology.**

***Index Terms*—Generative adversarial networks, Convolutional Neural Networks, ResNet, U-Net, Pix2Pix model.**

I. Introduction

Colors significantly influence our visual understanding of the environment, playing a fundamental role in shaping our perception of the world. As the saying goes, "Colors speak louder than words, expressing emotions, inspiring creativity, and painting the world with beauty." Image colorization involves the task of transforming a grayscale image, which represents low-dimensional data, into a high-dimensional representation by recovering its original colors. This technique finds applications in various domains, including color restoration and image colorization for animations, making it a challenging and exciting research problem due to its wide range of practical uses. Moreover, it is a multimodal problem since the same object can have different possible colors.

Deep generative modeling utilizes deep neural networks to learn a probability distribution over a given set of data points and generate similar data points. The field of generative modeling has seen significant advancements since the release of generative adversarial networks (GANs) in 2014. GANs consist of a generator and a discriminator, and variations such as Wasserstein GANs (WGANs) have been introduced to improve stability during training. Conditional WGANs (cWGANs) further extend the GAN framework by incorporating conditional information to guide the generation process.

This article aims to provide insights into cWGANs, which enhance the GAN framework by incorporating conditional information into both the generator and the discriminator.

†Department of Physics and Astronomy, Department of Information Engineering, University of Padova,
email: {name.surname}@studenti.unipd.it

Special thanks to Jacopo Pegoraro and Daniele Mari, Department of Information Engineering, University of Padova.

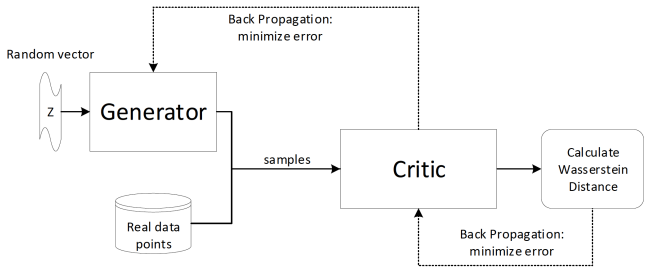

Fig. 1: WGAN

In cWGANs, an additional input, known as a condition, is provided to both the generator and the discriminator. This condition can be any auxiliary information that guides the generation process, such as class labels or attribute vectors. By conditioning the generation on specific information, cWGANs enable controlled and targeted synthesis of data samples. The discriminator in a cWGAN not only assesses the authenticity of the generated data but also considers how well it aligns with the provided condition, ensuring that the generated samples are realistic and consistent with the desired attributes or characteristics.

In this work, we aim to improve the Pix2Pix model, introduced by Isola *et al.* [1], by utilizing a Wasserstein GAN (WGAN) instead of the traditional GAN architecture. Furthermore, we employ a U-Net architecture based on residual blocks to enhance the performance of Pix2Pix. U-Net is a convolutional neural network (CNN) that excels in image segmentation tasks. It consists of convolutional and max pooling layers, with skip connections between layers of the same resolution. These skip connections enable the network to capture fine details of the input image, which is particularly beneficial for colorization tasks. Residual blocks, which comprise multiple layers with skip connections, allow gradients to propagate more effectively, aiding faster convergence and producing superior results. By combining WGAN with a U-Net architecture based on residual blocks, we can enhance the performance of Pix2Pix for colorization by providing better stability and improving the network's ability to capture fine details.

The proposed model takes grayscale images as input and generates corresponding colorized images.

In any deep learning task, including image colorization, data preparation is a crucial step. The input data for training our model should consist of black and white images, while the ground truth should provide color information. Although the RGB color space is the primary method for describing colors,

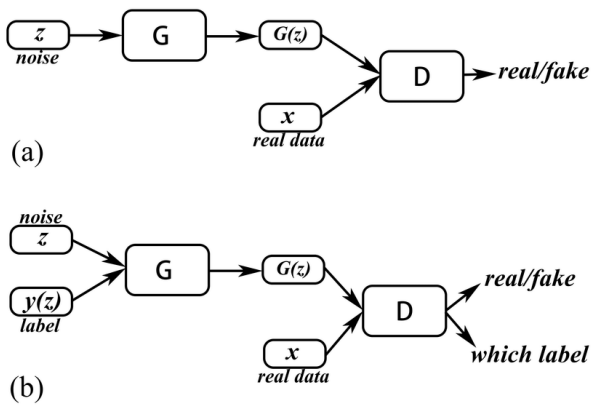

Fig. 2: (a) GAN (b) cGAN

other color spaces, such as HSV and CIE L*a*b (Lab), also exist. An image is represented as a batch of pixels, where each pixel is a 3-dimensional vector consisting of its height, width, and color. In the RGB color space, the color data is encoded by three values representing the intensity of Red, Green, and Blue channels, respectively. On the other hand, HSV (Hue, Saturation, Value) is a color model that characterizes colors based on their hue (dominant color), saturation (intensity or purity), and value (brightness). In the Lab color space, unlike RGB, the color and brightness of a pixel are represented by the three channels: ”L” (Lightness), ”a” (Red/Green), and ”b” (Blue/Yellow). In this work, we employ the Lab color space for training the model using deep learning. The initial grayscale image is considered as the L channel, and the objective is to reconstruct the two chrominance channels.

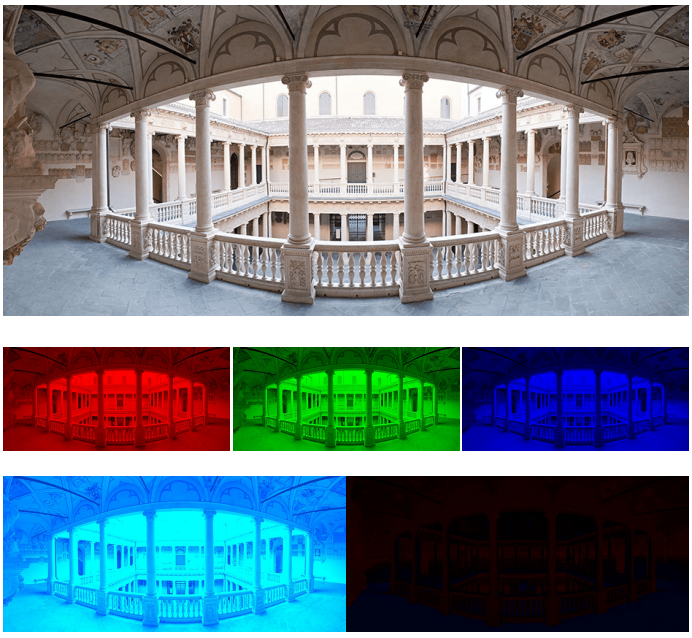

Fig. 3: The picture in the first row is a sample, images in the second row are the RGB channels, and L* and a*b* are shown in the last row.

II. Related Works

A. Automatic Image Colorization

The field of automatic image colorization has witnessed significant advancements with the emergence of deep neural networks and the improvement of processing units [2]. Many existing techniques utilize Convolutional Neural Networks (CNNs) as a key component for image colorization.

B. Generative Adversarial Networks

Generative Adversarial Networks (GANs) have revolutionized generative modeling by introducing an adversarial framework that simultaneously trains a generator and a discriminator [3]. GANs offer advantages such as eliminating the need for Markov chains, utilizing backpropagation for gradient computation, avoiding inference during learning, and accommodating a wide range of functions. However, GANs lack an explicit representation of the generator’s distribution and require synchronization between the generator and discriminator during training.

C. Conditional Generative Adversarial Networks

A notable work in the field is the conditional Generative Adversarial Network (cGAN) introduced by Isola et al. [1]. This approach incorporates conditional information, such as a target image, to generate new images. The Pix2Pix framework, based on cGANs, has been employed for various image-to-image translation tasks, including image colorization.

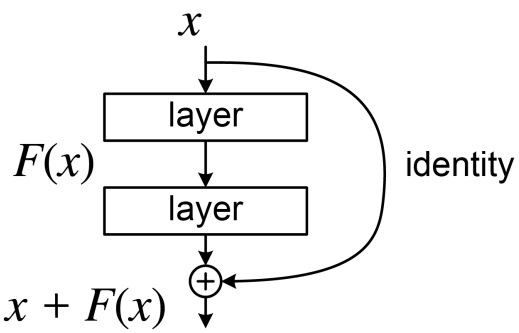

Fig. 4: ResBlock

III. Processing Pipeline

The proposed model is a CGAN (Conditional GAN) with the Wasserstein distance as the loss function for both generator and the discriminator. The Wasserstein distance is a measure of dissimilarity between two probability distributions. It is useful when dealing with distributions with different shapes, supports or dimensions. It considers the entire distribution as a hole and then quantifies the minimum cost of transforming one distribution into another, which is determined by the amount of ”mass” that needs to be moved and the distance it has to travel.

C-WGAN, or Conditional Wasserstein GAN, is an extension of the Wasserstein GAN (WGAN) that incorporates conditional

information into the generative adversarial network framework. It combines the Wasserstein distance as the training metric and the conditional generation capability of CGAN.

Generator

For constructing the generator, we implemented a ResU-Net architecture which is UNet with ResBlock for Semantic Segmentation. U-Net can be considered as an alternative to the sliding window convolution approach in certain contexts, especially when high-quality output is required. It is a convolutional neural network architecture designed explicitly for image segmentation tasks. Such a network can be trained end-to-end from very few images and outperforms other segmentation methods (a sliding-window convolutional network) [4].

ResNet, short for "Residual Network," is a deep learning architecture that was introduced by Kaiming He*et al.* [5]. Residual block(ResBlock) is the core concept of ResNet, which allow the network to learn residual mappings instead of directly learning the desired underlying mapping. A ResBlock is a fundamental building block in ResNet and also in a deep learning architecture, particularly in convolutional neural networks (CNNs). ResBlock with skip connections helped make a deeper and deeper convolution neural network and tackled the problem of degradation in deep networks, where the accuracy decreases as the network depth increases. It enables the flow of information directly through the network by utilizing skip connections or shortcuts. These skip connections allow the gradient to propagate effectively, enabling the training of very deep networks.

We split the generator's network into two parts, encoder, and decoder.

- The encoder captures features at different scales of the images by using a traditional stack of convolutional and max pooling layers. In fact, a block in the encoder consists of the repeated use of two convolutional layers ($k = 3$, $s = 1$), each followed by a non-linearity layer and a max-pooling layer ($k = 2$, $s = 2$). For every convolution block and its associated max pooling operation, the number of feature maps is doubled to ensure that the network can learn the complex structures effectively.
- The decoder path is a symmetric expanding counterpart that uses transposed convolutions. This type of convolutional layer is an up-sampling method with trainable parameters and performs the reverse of (down)pooling layers such as the max pool. Similar to the encoder, each convolution block is followed by an up-convolutional layer. The number of feature maps is halved in every block. Because recreating a segmentation mask from a small feature map is a rather tricky task for the network, the output after every up-convolutional layer is appended by the feature maps of the corresponding encoder block. The feature maps of the encoder layer are cropped if the dimensions exceed the one of the corresponding decoder layers.

The ReLU function is applied element-wise to the output of each convolutional layer in the residual blocks. ReLU is a non-linear activation function that introduces non-linearity to the model, allowing it to learn complex patterns and make the network more expressive. It sets all negative values to zero while leaving positive values unchanged. This helps in introducing non-linearities to the model and aids in learning complex features [6]. Specifically, our architecture, ReLU is used after each convolutional layer within the residual blocks (both in the encoding and decoding layers). It is applied to the intermediate feature maps, promoting sparsity and enabling the model to learn more discriminative representations. The ReLU activations help introduce non-linear transformations and increase the model's capacity to capture and model complex relationships within the data.

We used BatchNorm2d within the ResBlock and the encoding process modules after each convolutional layer. This helps in normalizing the activations, making the training process more stable and accelerating convergence also, allowing the network to learn and generalize better. By reducing internal covariate shift, BatchNorm enables higher learning rates and reduces the dependence of the network on specific weight initializations [7].

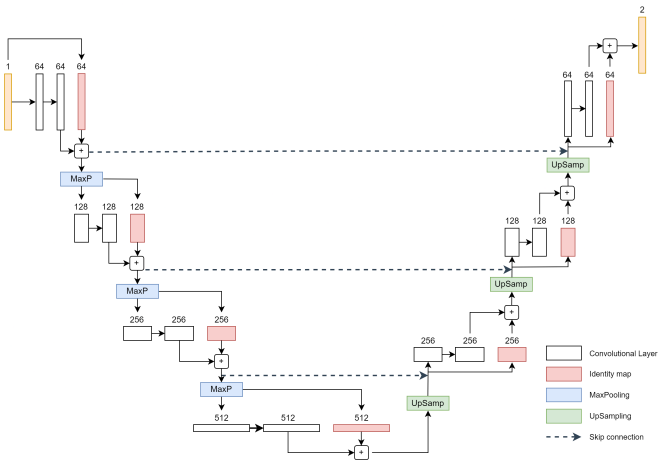

Fig. 5: U-Net with ResBlock

By using Dropout2d in ResBlock, and in the encoding process modules, we aimed to regularize the activations and prevent overfitting [8]. By setting the dropout rate to 0.2, 20% of the inputs were randomly set to zero.

For downsampling the input feature maps, we used the MaxPool2d layer of kernel size two and a default stride of two after the first convolutional layer in each DownSampleConv module. It applies a max pooling operation to the input tensor, reducing its spatial dimensions (width and height) while preserving the number of channels. By downsampling the feature maps, the MaxPool2d layer helps in capturing the most salient features while reducing the spatial information, which can be beneficial for learning hierarchical representations and extracting higher-level features [9].

Critic

In traditional generative adversarial networks (GANs), we have two main components: the Generator and the Discriminator. However, in Wasserstein GANs (WGANs), there is a common practice of referring to the Discriminator as the Critic, and we follow this convention. It is important to recognize that there is a subtle distinction between these components, and using the term "Critic" helps to clearly distinguish the WGAN architecture. The discriminator in a GAN is simply a classifier. It tries to distinguish real data from the data created by the generator. It could use any network architecture appropriate to the data type it's classifying. On the other hand, the Critic not only differentiates between real and fake images but rather estimates the Wasserstein distance between the distributions of real and generated data. This involves learning a value function that measures the quality of generated samples in terms of their similarity to real data, so the role of this component is not limited to binary classification.

The architecture of the critic network follows a standard convolutional neural network (CNN) design, where the input image is processed through a series of convolutional layers followed by batch normalization, LeakyReLU activation, and downsampling layers. The convolutional layers are designed to extract features from the input image, and the downsampling layers are responsible for reducing the spatial dimensions of the feature maps while increasing the number of filters. The architecture of the critic network is designed to handle the input images in the LAB color space, where the input image is the concatenation of the ab channels and the L channel. The output of the critic network is a scalar value, representing the probability of the input image being real or fake. The proposed architecture of the critic network is important for image recoloring task, as it allows the generator network to learn the underlying distribution of the real images in the LAB color space. By providing a reliable evaluation of the generated images, the critic network helps the generator network to produce more realistic and high-quality images. Additionally, the use of the LeakyReLU activation function and the Instance Normalization layers improve the performance of the critic network, as they help to stabilize the training process and reduce the mode collapse problem.

We set LeakyRelu's slope to 0.2 through the network to allow some information to pass for negative inputs. This addresses the "dying ReLU" problem, where ReLU units can become non-responsive to negative inputs during training [10]. Instance Normalization is an adaptation of Batch Normalization (BatchNorm) that normalizes the features within each instance (sample) independently instead of normalizing across the entire batch. This allows InstanceNorm to capture instance-specific statistics and improve the generalization and performance of the model [11].

Furthermore, the forward method takes in two inputs, "ab" and "l", concatenates them along the channel dimension, and passes them through the model. The model is a sequential of convolutional layers, Instance normalization, Leaky ReLU activation, AdaptiveAvgPool2d, Flatten and Linear layers. The output of the forward method is a single scalar value representing the Wasserstein distance between the true and fake data distributions.

The CWGAN (Conditional Wasserstein Generative Adversarial Network) uses a combination of loss functions to train the generator and the critic (discriminator) networks.

For the generator, the loss function is the reconstruction loss, which is calculated using the $L1$ loss (mean absolute error) between the generated images and the real images.

For the critic, there are three components to the loss function: Wasserstein loss: This is computed as the difference between the average scores given to real images and the average scores given to fake (generated) images. The critic aims to maximize this loss, while the generator aims to minimize it.

Gradient penalty: This penalty term encourages the Lipschitz constraint, which helps stabilize the training of the critic. It is computed by adding a gradient penalty term to the loss function, which penalizes the gradients of the critic's scores with respect to interpolated samples between real and fake images.

$R1$ regularization: This regularization term is used to enforce smoothness in the critic's output. It is computed as the squared $L2$ norm of the gradients of the critic's scores with respect to real images.

$$\begin{aligned} Loss_{cWGAN-GP} = \; & E_{z\sim P_z(z)}[D(G(z|y))] \\ & - \; E_{x\sim P_r}[D(x|y)] + \lambda \\ & . \; E_{\tilde{x}\sim P_{\tilde{x}}}[(\|\nabla_{\tilde{x}} D(\tilde{x}|y)\|_2 - 1)^2] \end{aligned}$$

The overall loss function of the CWGAN is a combination of these terms, where the generator aims to minimize the reconstruction loss, and the critic aims to maximize the Wasserstein loss while also considering the gradient penalty and $R1$ regularization. As mentioned before CWGAN uses a variant of the original GAN framework, called Wasserstein GAN (WGAN), which introduces the Wasserstein loss and gradient penalty to address some of the issues with traditional GAN training, such as mode collapse and training instability.

IV. Dataset and Features

Dataset

In this work, we use the MIRFLICKR25k dataset, which consists of 25000 colored images of different sizes collected from the popular photo-sharing platform Flickr. The dataset offers various images encompassing different categories, including nature, people, animals, buildings, and objects. We divided the dataset into two folders: one folder consisting of a and b dimensions of LAB color space images of the randomly sized colored image dataset and another folder consisting of a gray˙scale.npy file, which is the grayscale version of the dataset. Hence, grayscale images are taken as input for the model's training, and a and b components of LAB color space are taken as output.

Feature Extraction

Feature extraction is a fundamental aspect of conditional Wasserstein Generative Adversarial Networks (cWGANs) and is critical in determining the quality of the generated images. In cWGANs, the generator network produces synthetic data that closely resembles real data, while the discriminator network distinguishes between real and fake data. To achieve this, both the generator and discriminator utilize convolutional layers for feature extraction.

Convolutional layers are designed to learn local and hierarchical representations of the input data, allowing them to capture important features and patterns. These layers employ filters to detect specific visual characteristics such as edges, textures, and shapes within the input image. By convolving these filters across the image, convolutional layers create feature maps highlighting the presence and intensity of different features.

The extracted features are then propagated through the network, undergoing further transformations and combinations in subsequent layers. This enables the model to learn increasingly complex data representations, capturing high-level semantic information. As the training progresses, the generator learns to generate synthetic data that exhibits similar features to the real data, while the discriminator becomes more proficient at distinguishing between real and fake samples based on the extracted features.

Feature extraction in cWGANs is a crucial step that enables the model to learn the underlying distributions of the data and produce high-quality synthetic images that closely resemble the real data distribution.

V. Learning Framework

Proposed Method: The proposed method involves training a conditional Wasserstein GAN (cWGAN) to generate a colorized image from a grayscale input image. The generator, implemented as a Res-UNet, takes a random noise vector and the grayscale image as inputs to generate a colorized version of the image. The discriminator, a Convolutional Neural Network (CNN), takes both the original grayscale image and the generated colorized image along with conditioning information and classifies them as real or fake.

Feature Extraction: During training, the discriminator learns to distinguish between real and generated images by developing an internal representation of the distinguishing features. This representation acts as a feature extractor, capturing and representing the important characteristics of the input images.

Colorization: The MIRFLICKR25k dataset is used for training and evaluation in this project. The model is trained for different numbers of epochs, ranging from 20 to 80. The results section will demonstrate the impact of increasing the number of epochs on the generator and critic losses.

Model Evaluation: The "Wasserstein GAN with Gradient Penalty (WGAN-GP)" loss is employed in this project. The WGAN-GP loss is a variant of the original GAN loss that addresses limitations such as mode collapse and training instability. It comprises two key components:

1. Adversarial Loss: The adversarial loss drives the adversarial training between the generator and discriminator networks. The generator aims to generate realistic colorized images to deceive the discriminator, while the discriminator aims to distinguish between real color images and generated color images. This loss encourages the generator to improve its ability to produce realistic images.

2. Gradient Penalty: The gradient penalty is specific to WGAN-GP and enforces the Lipschitz constraint on the discriminator. It penalizes the gradients of the discriminator with respect to interpolated samples between real and generated images. The gradient penalty promotes smoothness in the discriminator's output and enhances training stability.

VI. Results

We trained our model on the MIRFLICKR25k dataset, fixing the input image size to 224×224 pixels. The training process utilized the Adam optimizer and PyTorch software packages accelerated on GPU hardware (NVIDIA T4 x2 GPU available on Kaggle).

The training process of our proposed method is as follows:

1) Initialize the generator network (G) and the critic network (D) with random weights.
2) Optimize the Wasserstein distance between the distribution of real images and generated images by iteratively updating the weights of the generator and critic networks using gradient descent.
3) Introduce a gradient penalty term to enforce the Lipschitz constraint on the critic network, ensuring training stability.
4) Choose the number of training iterations (epochs) and batch size.
5) For each training iteration, perform the following steps:
   - Sample a batch of real data samples and their corresponding conditioning information.
   - Generate a batch of fake samples by passing random noise and the conditioning information through the generator.
   - Update the critic network by computing the critic loss and taking a gradient step to minimize the loss.
   - Calculate the gradients of the critic's output with respect to interpolated samples between real and fake samples, and enforce the gradient penalty during the update.
   - Update the generator network by computing the generator loss and taking a gradient step to minimize the loss.

Epoch	Generator Loss	Critic Loss
1	0.032473	1.892919
20	0.030137	1.682347
40	0.030648	2.719594
60	0.026841	2.283246
80	0.0287426	2.650611

TABLE 1: Generator and Critic Losses for different epochs.

Fig. 6: Input data (left), real image (middle), and generated output at epoch 1 (right).

Fig. 7: Input data (left), real image (middle), and generated output at epoch 20 (right).

The generator's loss in Table 1 shows that the minimum loss occurs at epoch 60, while the critic loss is highest at epoch 40. Therefore, the optimal overall loss should be between epochs 40 and 60.

Furthermore, Fig. 8 and Fig. 9 showcase more realistic generated images for epochs 40 and 60, respectively.

VII. Concluding Remarks

This work aimed to build a deep learning-based approach for handling the automatic colorization of grayscale images. The combination of a WGAN and a U-Net generator with ResBlock can provide good results by utilizing the adversarial training of the cGAN and the encoding-decoding structure of the U-Net generator. The U-Net generator, allows the network to effectively capture the high-level features and the low-level details of the image, making it a good choice for image-to-image translation tasks such as colorization. However, itâ€™s important to note that the success of the method depends on the training data quality and the network architecture design. In the project we have been discussing, the model is based on the Wasserstein Generative Adversarial Network (WGAN) framework instead of the original pix2pix architecture. WGAN is a variant of the traditional Generative Adversarial Network (GAN) that introduces the Wasserstein distance as a more stable and meaningful metric for training the generator and discriminator networks. The WGAN architecture consists of a generator network and a critic network. The generator takes a grayscale image as input and generates a colorized output, similar to the pix2pix model. The critic network acts as the discriminator and distinguishes between real and generated images.

VIII. Future Works

Due to the constraints of limited time and computational resources, we encountered significant limitations that constrained our ability to explore and implement various untested ideas for further enhancing this work.

Fig. 8: Input data (left), real image (middle), and generated output at epoch 40 (right).

Fig. 9: Input data (left), real image (middle), and generated output at epoch 60 (right).

- Run The program on more powerful machines. Because of our low computation power, we had to run the network for small epochs and rely on the pre-made results. Three epochs with only 25 percent of our dataset take around 3 hours on the M1 chip. Therefore, we are going to run the network by cloud Veneto with high computational power.
- Run the program on different datasets. We ran our model on the MIRFLICKR25k, and we are going to run the model on other colorful datasets like ImageNet and COCO. ImageNet consists of 14 million labeled images that cover a wide range of object categories, including animals, objects, scenes, and more. However, MIRFLICKR25k has only 25000 user-generated content from Flickr.
- Try different activation functions. Although the model is well designed and the activation functions are running perfectly, there is always room for improvement. We are going to use more advanced optimizers like Swish and ELU. Surly the best choice of activation function depends on the specific characteristics of the data and the performance we observe through experimentation.
- Test different optimizers. Trying advanced optimizers like AdaDelta, Nadam, and Adamax could produce more reliable results. Indeed, it is important to monitor the training progress, analyze the loss curves, and evaluate the quality of colorized images to determine the effectiveness of the chosen optimizer.

IX. Challenges and Learning

Our comprehension of Generative Adversarial Networks (GANs) can be divided into two distinct phases, pre and post-this project. This project's successful completion necessitated acquiring knowledge in avant-garde technology and relatively novel mathematical paradigms.

Fig. 10: Input data (left), real image (middle), and generated output at epoch 80 (right).

In the implementation part, the primary challenge encountered was the implementation of PyTorch, which was initially challenging due to our lack of prior experience with the platform. A persistent issue we grappled with was the "dimension error" in PyTorch, a hurdle that remained until the completion of the project. Using tensors and pre-processing the dataset for model comprehension consumed a significant portion of our development time. Through our engagement with this framework, we discovered intriguing modules. Consequently, we have resolved to employ PyTorch for future image processing tasks instead of TensorFlow. Additionally, we expanded our knowledge base with various datasets in image processing. The existence of extensive datasets created by research laboratories highlights the significance and potential of this field.

Beyond the practical implementation, the mathematical constructs presented another fascinating project fact. The introduction to the Wasserstein metric revolutionized our perspective on GANs. Subsequently, integrating a conditional setting with the Wasserstein GAN (WGAN) emerged as another concept that demanded substantial research time. The ResU-Net architecture, particularly its skip connections, was another concept we mastered during the project implementation. Furthermore, we started the study of different color spaces, an area of knowledge we expanded upon during this project. Initially, the Lab color space and its implications were challenging to comprehend, but we quickly grasped its significance and incorporated it into our project.

In conclusion, we have decided to participate in the Kaggle competition " I am Something of a Painter Myself Use GANs to create art" to apply our newly acquired knowledge in a non-academic setting. After that, surely, we will add PyTorch. Moreover, our research revealed that GANs have potential cybersecurity applications, such as anomaly and malware detection, beyond image processing. This could be useful for us in Network and Security course the next semester.

REFERENCES

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2018.

[2] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, "Real-time user-guided image colorization with learned deep priors," *ACM Trans. Graph.*, vol. 36, jul 2017.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

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

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[6] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*, 2010.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[10] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[11] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2017.