Image Colorization using Conditional Generative Adversarial Networks

Mojtaba Roshana†, Roya Joulaei Vijouyeh †, Asal Rangrazi Asl†

***Abstract*—Over the last decade, the process of automatic image colorization has been of significant interest for several applications and Recently Deep Neural Networks have shown remarkable success in automatic image colorization. The paper presents a method for colorizing grayscale images using a conditional Generative Adversarial Network (cGAN) based on the UNET architecture. The cGAN is trained to generate color pixels for the grayscale image by taking both the grayscale image and a random noise vector as inputs. In our approach, we develop a Pix2Pix conditional GAN network using Patch-GAN that operates on image patches, rather than on the whole image. The designed network will implement on datasets ImageNet-1000 for adding color information to a grayscale image to produce a full-color image .**

***Index Terms*—Pix2Pix model , Generative Adversarial Network , Neural Networks , Patch GAN , UNET Autoencoders, pre-trained Resnet18,fine-tuning**

I. Introduction

Color plays a key role in the process of human cognition of the world and rich colors can not only express more information but also enhance the human visual experience. Image colorization has been a very active research topic in the field of digital image processing, and is an inter-disciplinary area involving disciplines such as Computer Vision, Computer Graphics, Pattern Recognition, and Human-Computer Interaction. Colorization has been widely used in many fields, such as grayscale photo colorization, old film color restoration, cartoon automatic colorization, etc. The automatic colorization of grayscale images has been an active area of research in machine learning for an extensive period of time. In this manuscript, we will explore the method of colorization using Conditional GAN (cGANs) proposed by Goodfellow et al. [1]. In deep generative modeling, deep neural networks learn a probability distribution over a given set of data points and generate similar data points. Since it is an unsupervised learning task it does not use any labels during the learning process. Since its release in 2014, the deep learning community has been actively developing new GANs to improve the field of generative modeling. This article aims to provide information on cGANs, specifically Pix2Pix GAN, which is one of the most used generative models.

†Department of Physics and Astronomy, University of Padova, email:{mojtaba.roshana,roya.joulaeivijouyeh,asal.rangraziasl}@studenti.unipd.it

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

Image colorization: In image colorization, the first step is to prepare our data. The data that is required to train our model with should be black and white, whereas the validation set (the ground truth) should be in colors. [2]
RGB and CIE L*a*b* (or just "Lab") are two different color spaces or ways of describing colors. When we load an image, we get a rank-3 (height, width, color) array with the last axis containing the color data for our image. These data represent color in RGB color space or LAB space. In RGB color space there are 3 numbers for each pixel indicating how much Red, Green, and Blue the pixel is. However, in LAB color space, we have again three numbers for each pixel but these numbers have different meanings. The first number (channel), L, encodes the Lightness of each pixel, and when we visualize this channel (the second image in the row below) it appears as a black-and-white image. The a and b channels encode how much green-red and yellow-blue each pixel is, respectively. In Figure 1 you can see each channel of Lab color space separately. [3]

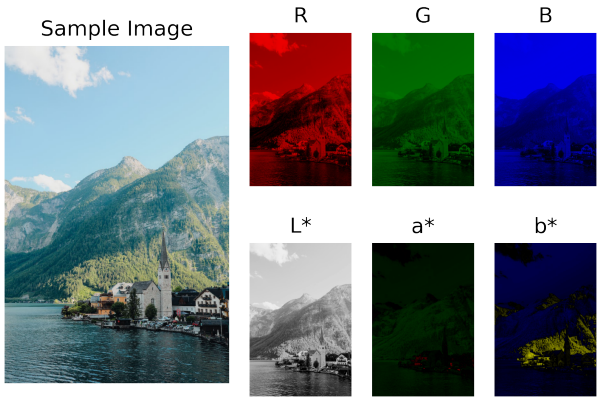

Fig. 1: Comparison of RGB and Lab adjustments

In this paper we use Lab color space instead of RGB to train the models because to train a model for colorization, we should give it a grayscale image and hope that it will make it colorful. When using Lab, we can give the L channel to the model (which is the grayscale image) and want it to predict the other two channels (a, b) and after its prediction, we concatenate all the channels and we get our colorful image.

II. Related Work

With the great progress of deep learning in image processing, deep neural networks' training and the mapping of grayscale images to color images are achieved by using

largescale grayscale images and corresponding color images, to complete the grayscale image colorization.

”Colorful Image Colorization” by Zhang et al. (2016) [4] is a seminal work in the field of deep learning-based image colorization, where the authors proposed a fully convolutional neural network for the task.

”Perceptual Losses for Real-Time Style Transfer and Super-Resolution” by Johnson et al. (2016) [5] presents a method for training neural networks to generate high-quality images, including colorization, using a perceptual loss that measures the difference between the generated image and the ground truth in terms of human perception.

”Let There Be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification” by Cheng-Lin Liu, Weilin Huang, Ning Zhang, and Ming-Hsuan Yang (2018) [6] proposes a method for colorization that combines global and local image priors to generate more accurate colorizations.

”Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2” by M. O. SÃ¸rensen and J. R. Ã˜sterby (2018) [7]presents a deep learning-based method for image colorization that uses a combination of convolutional neural networks and the Inception-ResNet-v2 architecture.

In this paper, We are focused on the Image-to-Image Translation with Conditional Adversarial Networks by Zhu et al. (2017) [8] which explores GANs in the conditional setting. This method also differs from the prior works in several architectural choices for the generator and discriminator. Unlike past works, we use a UNet-based architecture for the generator and using a convolutional PatchGAN for the discriminator. In the end, we will also use transfer Learning-Based Colorization which is a method that involves using pre-trained deep neural networks, such as ResNet, to colorize images and compare the results.

III. Processing Pipelin

The proposed methods structure is represented in Figure 2. We implement Conditional Generative Adversarial Networks (cGANs) which are a type of generative model that has been applied to the problem of image colorization, among other tasks. In this case, the cGAN takes a grayscale image as input and generates a corresponding colorized image.

In a cGAN architecture, for constructing the generator network of the pix2pix we will use UNET which is a U-shaped encoder-decoder network architecture followed by the ReLU activation function. All ReLUs in the encoder are leaky, with the slope=0.2, while ReLUs in the decoder are not leaky. Most of the convolutional layers will also be followed by the batch normalization layers(except for the first C64 layer). Once we return the encoder block of the generator network, we can construct the first half of the network as follows:

C64-C128-C256-C512-C512-C512-C512-C512

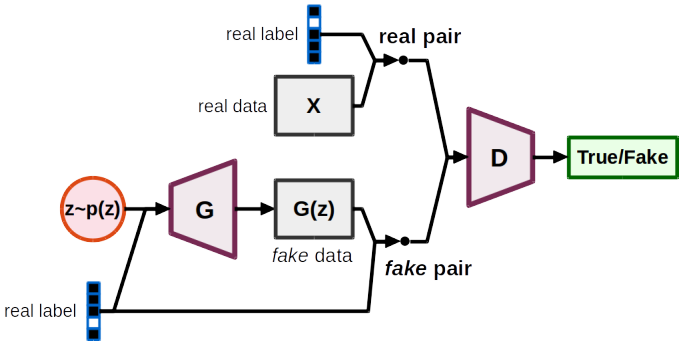

Fig. 2: conditional GAN architecture

The next part of the generator network we will define is the decoder block. In this function, we will upsample all the previously downsampled images as well as add (concatenate) the necessary skip connections that must be made from the encoder to the decoder network, similar to the U-Net architecture. For the upsampling of the model, we can make use of the convolutional transpose layers with batch normalization layers and optional dropout layers. dropout is used in GANs as a regularization technique to prevent overfitting, improve the stability of the training process, and encourage the learning of more robust features. In this project, we use dropout only in Generator as roles of noises which will be explained. The decoder block of the generator network contains the architecture as follows:

CD512-CD512-CD512-C512-C256-C128-C64

The U-Net architecture (see Figure 3) is identical except with skip connections between each layer i in the encoder and layer n-i in the decoder, where n is the total number of layers. The skip connections concatenate activations from layer i to layer n-i. This changes the number of channels in the decoder: U-Net decoder:

CD512-CD1024-CD1024-C1024-C1024-C512 -C256-C128

After the last layer in the decoder, a convolution is applied to map the number of output channels (2 channels in this task). We use the hyperbolic tangent (tanh) activation function in the last layer of a U-Net architecture because it has a range of -1 to 1, which is convenient for probability-like predictions, and also due to its smoothness and non-linearity.

the generator network takes two inputs: a random noise vector (z) and a condition, which in the case of image colorization is the grayscale image(x). The generator(G) network then uses these inputs to generate a colorized image.

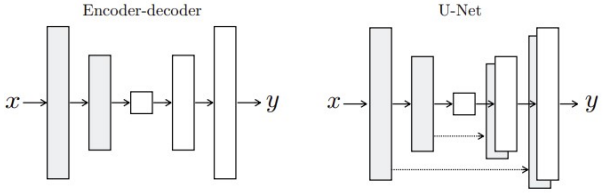

Fig. 3: Illustration of UNET and Encoder-Decoder architecture

For constructing the discriminator we use patch GAN which takes in both the grayscale image and colored image and evaluates whether the colored image is a valid transformation of the grayscale input. The PatchGAN has an effective receptive field of 70x70. every single cell in the output array tells the probability of a 70x70 patch being real or fake (see Figure 4). The discriminator structure will follow the pattern build of C64-C128-C256-C512. We implement a model by stacking blocks of Conv-BatchNorm-LeackyReLU to decide whether the input image is fake or real. Besides, the first and last blocks do not use normalization and the last block has no activation function(It is embedded in the loss function).

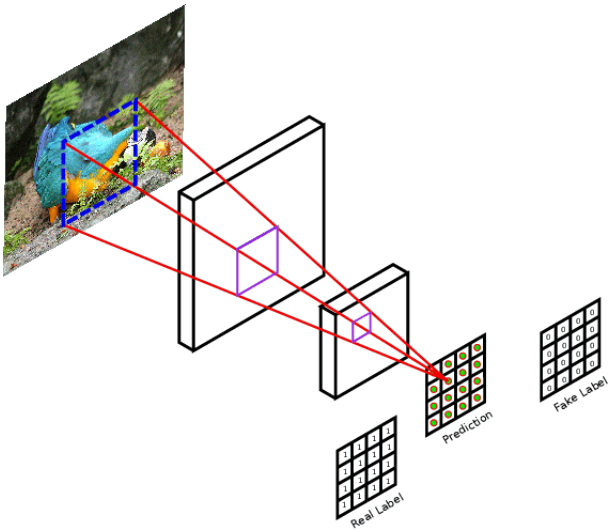

Fig. 4: Illustration of PatchGAN

To optimize our networks, we alternate between one gradient descent step on Discriminator, then one step on Generator. We use minibatch SGD and apply the Adam solver with a learning rate of 0.0002, and the momentum parameters of β_1 = 0.5, β_2 = 0.999 [8], [9]. We use Adam optimizer due to its simplicity, computational efficiency, and good performance. The two networks are trained together in an adversarial manner, with the generator trying to produce images that are indistinguishable from real color images and the discriminator trying to correctly classify the generated images as fake. Over time, the generator improves its ability to generate realistic color images, and the discriminator improves its ability to distinguish between real and fake images. The generator loss is usually defined as the negative log-likelihood of the discriminator's output, given the generated sample and the conditioning input. The generator tries to minimize this loss, which encourages it to generate samples that are classified as real by the discriminator. However, the discriminator loss measures the binary cross-entropy between the true labels (real or fake) and the discriminator's predictions. The discriminator tries to maximize this loss, which encourages it to produce accurate predictions.

The loss functions for a standard Generative Adversarial Network (GAN) and a Conditional Generative Adversarial Network (cGAN) are similar, but with some important differences. The cGAN loss function includes the additional condition(Label) as input to both the generator and discriminator, allowing the cGAN to take into account the additional information specified by the condition when learning to generate new images. Here you can see the loss formula for both cGAN, GAN, and L1. The L1 alone leads to reasonable but blurry results. The cGAN alone(λ= 0) gives much sharper results but introduces visual artifacts on certain applications. Adding both terms together(λ = 100) reduces these artifacts. [10]

Loss formula for GAN :

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

Loss formula for cGAN :

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x,G(x,z))], \tag{2}$$

loss of L1:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}\left[\|y - G(x,z)\|_1\right] \tag{3}$$

Our final objective is :

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \tag{4}$$

For our final models, we provide noise only in the form of dropout, applied on several layers of our generator at both training and test time. Despite the dropout noise, we observe only minor stochasticity in the output of our nets. Designing conditional GANs that produce highly stochastic output, and thereby capture the full entropy of the conditional distributions they model, is an important question left open by the present work.

IV. Dataset and Features

Dataset: The dataset we are using for this project is basically a sample of the well-known and substantial image-net dataset. Image-net is one of the biggest visual datasets available nowadays. ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset". There are more than 100,000 synsets in WordNet; the majority of them are nouns (80,000+). The dataset contains millions of images that have been labeled for supervised learning purposes but in this case, we don't need the labels. The input of colorization is a noncolored photo and the output is the colored one. The "image-to-image translation with conditional adversarial network" paper uses the whole ImageNet dataset which contains 1.3 million images. In this project, we trained and evaluated our models using 5000 samples from it.

We assume 80% (4000 images) of data as the train set and 20% (1000 images) as the validation set.

Feature extraction: Feature extraction in cGANs for colorization is a crucial step because it determines the quality of the generated images. The generator is trained to produce data that is indistinguishable from real data, and in the process, it learns the underlying distributions of the data and extracts the important features. The discriminator, on the other hand, is trained to identify fake data and, in the process, it learns to recognize the important features of the real data. To extract features, the generator and the discriminator networks typically use convolutional layers, which are designed to learn local and hierarchical representations of the input image. The convolutional layers use filters to identify features such as edges, textures, and patterns in the input image. These features are then passed through multiple layers of the network, where they are transformed and combined to generate the final output image.

V. Learning Framework

Proposed Method: The model is to train a cGAN to generate a colored image from a grayscale input image by implementing UNET as a generator and PatchGAN for the discriminator part. the generator network takes two inputs: a random noise vector and the grayscale image then uses these inputs to generate a colorized version of the input image. The discriminator network takes in both the original grayscale image and the generated colorized image and attempts to classify the generated image as either real or fake.

Feature Extraction: During training, the discriminator sees both real images and generated images, and it develops an internal representation of the features that distinguish real images from fake ones. This internal representation can be thought of as a feature extractor, as it extracts and represents the important features of the input images.

colorization: In this project, we trained and evaluated our models using 5000 samples from the dataset. After splitting the data, we trained our model for 45 epochs. In the result part we show how increasing numbers of epochs affected directly the output and step by step the gray inputs become colorful.

Pre-trained Model: Using a pre-trained model for classification as the backbone of the generator's down-sampling path in a cGAN for colorization can be a good choice as it can provide a strong foundation of features to build upon. When using a pre-trained model, the model has already learned to recognize various visual features from large amounts of data. These features can be used as a starting point for the generator in a cGAN, allowing it to quickly learn the mapping between grayscale images and color images. This can be particularly useful when the training data set for colorization is small or limited. The pre-trained model can provide a rich set of features that would be difficult to learn from scratch. It is important to keep in mind that the pre-trained model was designed for a different task (classification) and its features may not be optimal for colorization, so we add fine-tuning of the pre-trained model to adapt it to the colorization task by removing the last two layers (GlobalAveragePooling and a Linear layer for the ImageNet classification task) and uses this backbone to build a U-Net with the needed output channels (2 in our case) and with an input size of 256x256.

Model Evaluation(losses): we use the "vanilla" GAN loss in our project and register some constant tensors as the "real" and "fake" labels. Then when we call this module, it makes an appropriate tensor full of zeros or ones (according to what we need at the stage) and computes the loss. Also, we use L1 loss as well and compute the distance between the predicted two channels and the target two channels and multiply this loss by a coefficient which in our case($\lambda = 100$) to balance the two losses and then add this loss to the adversarial loss. Then we call the backward method of the loss.

VI. Results

We have trained our model on the ImageNet-1000 dataset. We randomly selected a subset from the dataset without any constraint on the category of the images so that the network would not be biased toward specific color tones. We fixed the input image size for all models to 256-256 pixels. The Adam optimizer is used for the training process and all models are trained using PyTorch software packages accelerated on GPU Hardware (NVIDIA T4 Tensor Core GPUs google colab). The training process of our proposed method is organized as follows. first, we make the Evaluation of the main model which has been trained by the ImageNet-1000 dataset for 45 epochs with 4000 training data and 1000 validation. Also, we did a random rotation on training images to add a little data augmentation.

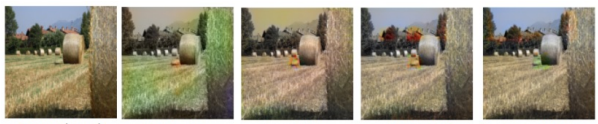

Fig. 5: Evaluation of main model for 45 epochs

In Figure 5 we can realize that in a cGAN, the generator and discriminator networks are trained over multiple epochs to learn the mapping between grayscale images and color images. Increasing the number of epochs can allow the network to learn more complex relationships and generate more accurate colorization. By training the model for more epochs or more data, the results will be more realistic. In this project, we only could run 45 epochs because of limited resources. The results of the training model in epoch 45 are shown in Figure 9. The model has been trained to distinguish some objects and colorize them. But for small objects and details results are not good enough. To measure the accuracy of colorization we need human test which means that the human brain should not be able to discriminate between real and generated images. In Figure 9 we can easily realize that some of the generated images are not real. Also, it is hard to

guess that two of generated images (third and fourth image from left to right second row) are fake or real. Therefore by training this model in more epochs and also by increasing the data we can get better results. In the following, we tried a pertained ResNet18 model which gives better results.
During the implementation of Conditional GAN for 45 epochs, we calculate the GAN losses of our final model. Moreover, to monitor the learning process of our models, model losses were plotted with respect to the number of iterations. In Figure 6 we can see the discriminator losses which contain fake loss and real loss. By averaging the fake and real loss we can get the loss of the discriminator. Also, we can see that the loss_G_GAN which we defined in equation (2) is reduced when the number of iterations is increased.

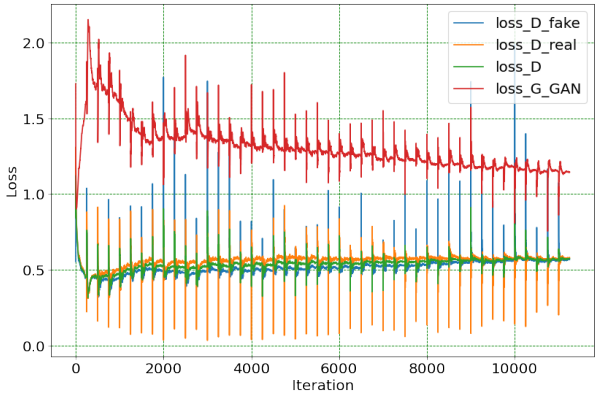

Fig. 6: The Discriminator and G_GAN loss for 45 epoches

In Figure 7 It can be seen that by increasing the number of iterations the loss_G ,which we defined in equation (4), is getting close to loss_D. Besides, in Figure 8 we can see the final gan and discriminator loss. Due to the limited resources mentioned before, it was not possible to train the model for more epochs. However, we can realize the reduction of the GAN loss after 4000 iterations which is equal to epoch 16.

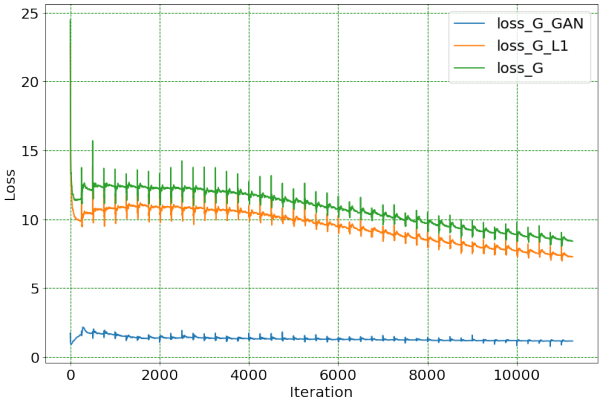

Fig. 7: The Generator loss for 45 epoches

Testing the COCO (Common Objects in Context) dataset for colorization can be a good way to evaluate the performance of the cGAN for colorization. In this section, we test our

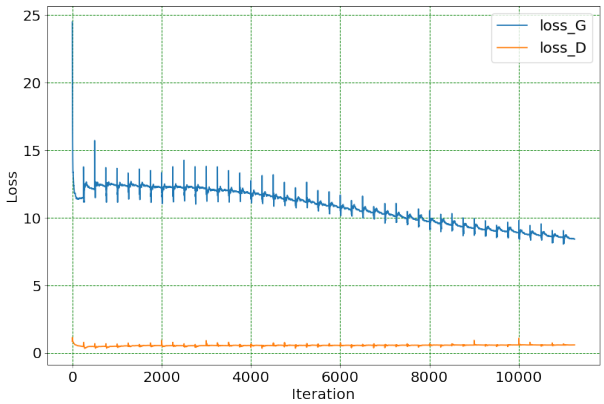

Fig. 8: Generator and Discriminator losses for 45 epoches

Fig. 9: Training our model in epoch 45 with 4000 training data and 1000 validation. The first row is grayscale, the second row is generated image by the model, and the third row is a ground truth image

Fig. 10: Test the pretrained model (using ResNet18) after training 20 epochs. The first row is grayscale, the second row is generated image by the model, and the third row is a ground truth image

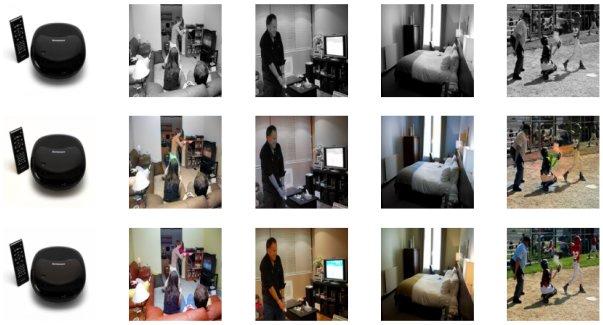

Fig. 11: Test our model with the coco dataset. The first row is grayscale, the second row is generated image by the model, and the third row is the ground truth image

model with the coco dataset which gives good results. We should notice that this data set is quite similar to ImageNet (see Figure 11).

As we mentioned before we use a pre-trained ResNet18 as the backbone of our U-Net. In Figure 10 you can find that it quickly learns the mapping between grayscale images and color images. As a result, using a pre-trained model as the backbone of the generator's down-sampling path in a cGAN for colorization can be a useful approach. In fact, we use a pre-trained ResNet18 as the backbone of the U-Net and to accomplish the second stage of pretraining, we train the U-Net on our training set with only L1 Loss in 20 epochs. Then we move to the combined adversarial and L1 loss, as we did in the previous approach, and trained it for 20 epochs which we can see the losses in Figure 12. By comparing Figure 12 and 8 we can get that the pre-trained model has better performance because in fewer number iterations gets less loss.

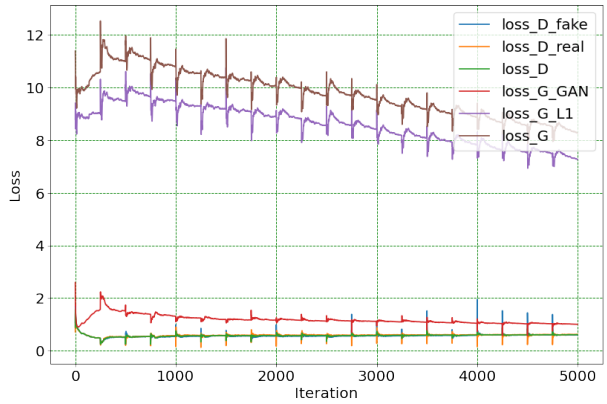

Fig. 12: Losses of generator and discrimination after 20 epochs in the pre-trained model (using ResNet18)

We find the same model which has been trained with 10000 COCO images with more training epochs. Last but not least, we did interesting things by giving random images as input for the model to test it.

Part 1: we selected four old grayscale pictures to make them colorful for testing the model performance (see Figure 13).

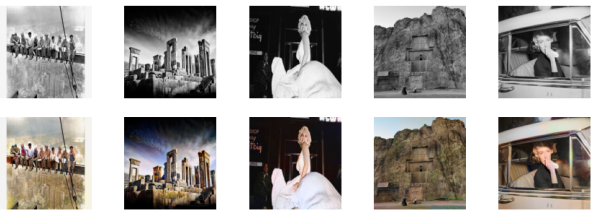

Fig. 13: Pretrained model (using ResNet18) results for real black and withe images

Part 2: Giving the input images of our daily life images to test the model. All the images were taken by ourselves from the story of our student life at the university of the Padova (see Figure 14).

Fig. 14: Use the pre-trained model (using ResNet18) to apply on some photos of our daily life. The first row is grayscale, the second row is generated image by the model, and the third row is a ground truth image

VII. Concluding Remarks

The goal of this work was to build a deep learning-based approach for handling the automatic colorization of grayscale images. The combination of a cGAN and a U-Net generator 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, on the other hand, 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, it's important to note that the success of the method depends on the quality of the training data and the design of the network architecture.

References

[1] G. Perarnau, J. Van De Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional gans for image editing," *arXiv preprint arXiv:1611.06355*, 2016.

[2] M. W. Schwarz, W. B. Cowan, and J. C. Beatty, "An experimental comparison of rgb, yiq, lab, hsv, and opponent color models," *Acm Transactions on Graphics (tog)*, vol. 6, no. 2, pp. 123–158, 1987.

[3] J. An, K. G. Kpeyiton, and Q. Shi, "Grayscale images colorization with convolutional neural networks," *Soft Computing*, vol. 24, pp. 4751–4758, 2020.

[4] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *European Conference on Computer Vision*, 2016.

[5] J. Johnson, A. Alahi, and F.-F. Li, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, pp. 694–711, Springer, 2016.

[6] C.-L. Liu, W. Huang, N. Zhang, and M.-H. Yang, "Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8411–8420, IEEE, 2018.

[7] M. O. Sørensen and J. R. Østerby, "Deep koalarization: Image colorization using cnns and inception-resnet-v2," in *International Conference on Computer Vision Theory and Applications*, pp. 408–415, SciTePress, 2018.

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2017.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. 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.