”Standard GAN vs WGAN” for image colorization

Elena Zoppellari†, Valentina Tonazzo†

***Abstract*—In this report, the authors present a comprehensive study on the implementation of a Generative Adversarial Network (GAN) for image colorization. The aim of this study is to improve the quality of image colorization by adapting the GAN structure specifically for this task and comparing its performance with a Wasserstein-GAN (WGAN) structure. The experiments were conducted on the Tiny-ImageNet-200 dataset, a reduced version of the ImageNet-1000 dataset, offering a unique perspective on the challenges of using an uncommon dataset for this task. The contribution of this study is multi-faceted. Firstly, it showcases the versatility of GANs, demonstrating their ability to be adapted to various image processing tasks, such as image colorization. Secondly, it provides a comparison between the performance of a WGAN and a simple GAN, highlighting the strengths and weaknesses of each model. This paper aims to offer a new perspective on the use of GANs for colorization and provides valuable insights for future research in this field.**

***Index Terms*—Neural Network, Image colorization, Generative Adversarial Networks, Wasserstein GANs, Autoencoders, Wasserstein distance, Wasserstein loss.**

I. Introduction

Image colorization is the process of adding colors to grayscale or black and white images. This challenging problem involves assigning colors to each pixel in the image, and requires an understanding of the semantic meaning and relationships between colors in a particular scene. The primary objective of colorization is to produce a visually appealing and realistic color image that accurately represents the original grayscale image. A successful algorithm must generate accurate and plausible color distributions, maintain image details and structures, and effectively handle complex lighting and shading effects.

In recent years, Convolutional Neural Networks (CNNs) have been widely used to tackle the problem of image colorization. The objective of these networks is to minimize the Euclidean distance between the predicted and actual pixels, but this approach often leads to poor results such as blurred images and lack of vividness. Another challenge with this approach is the loss function, which requires human expertise to design appropriately. Some progress has been made with generative models, but these can still produce low-quality samples or fail to converge, leading to unstable training.

As a first step, this project implements a Neural Network designed to learn a loss that classifies the output colored image as real or fake while simultaneously training a generative model to minimize this loss. This type of network is called a Generative Adversarial Network (GAN), and it has been widely studied in recent years for a variety of tasks such as image recognition, image segmentation, and general image-to-image translation. The primary goal of this project is to apply this structure to the task of image colorization.

In order to improve the stability and realism of the model, the project then explores whether the discriminator model can be improved using a Wasserstein-GAN (WGAN) structure, which is state-of-the-art in this field, and compares the results against a simple GAN model. All of the work was done on the Tiny-ImageNet-200 dataset, which is a reduced version of the ImageNet-1000 dataset. This dataset presents a new challenge: while it allows for faster training and testing, the reduced amount of information contained in the dataset makes it harder for a Neural Network to extract meaningful information.

In conclusion, the report makes the following contributions:

- The adaptation of a Generative Adversarial Network (GAN) structure for the task of image colorization.
- A comparison of the performance of a Wasserstein-GAN (WGAN) structure versus a simple GAN structure.
- An exploration of the use of the non-standard dataset, Tiny-ImageNet-200, for the task of image colorization.

The report is structured as follows: Section II provides an overview of the related works presented in literature. Section III outlines the organization and main steps of our work. Section IV describes the dataset and preprocessing efforts. Section V delves into the details of the implemented models. Section VI presents the results obtained from our experiments. Finally, in Section VII, the conclusions of our work are presented.

II. Related Work

As previously noted, Convolutional Neural Networks (CNNs) have been widely used in many image processing problems in recent years. One notable example is the U-Net Convolutional Neural Network, which won the ISBI cell tracking challenge 2015 for Image Segmentation. Ronneberger et al. [1] proposed a novel structure that consists of an autoencoder model, where the output of the encoding part is concatenated with the input of the decoding part to preserve information such as positional location and semantic meaning.

In [2], Isola et al. utilized a Generative Adversarial Network (GAN) structure to address a variety of Image-to-image translation problems and achieved impressive results. The generative model was implemented as a U-Net-based

†Department of Physics and Astronomy, University of Padova, email:{elena.zoppellari}@studenti.unipd.it, {valentina.tonazzo.1}@studenti.unipd.it

autoencoder Neural Network, while the discriminator model consisted of a Conditional GAN structure referred to as a convolutional "PatchGAN" classifier, which only penalizes structures at the scale of image patches.

Later, Arjovsky et al. made an improvement from a theoretical point of view and an interesting computational proposal in [3], where they defined a form of GAN called Wasserstein-GAN that minimizes an efficient approximation of the Wasserstein distance, which represents the "cost" of the optimal transport plan in probability distribution functions.

The tiny-ImageNet-200 classification challenge is similar to the classification challenge in the full ImageNet ILSVRC. tiny-ImageNet-200 contains 200 classes for training, with 500 images per class. The test set contains 10,000 images, all of which are 64x64 colored images. This dataset was first used in [4] for a visual recognition challenge.

III. Processing Pipeline

The computational steps behind the study are described as follows:

- **Data acquisition:** As mentioned before, our model has been trained on Tiny ImageNet, which is a subset of the ImageNet dataset. This choice was made to strike a balance between computational availability and training the model on a dataset with sufficient variability.
- **Images pre-processing:** Each image in the dataset is represented in the RGB color space, so it has a dimension of $64 \times 64 \times 3$ where 3 is the number of channels (red, blue, and green), which combine additively to determine the color of the image. As explained in more detail in the next section, we decided to convert the images from the RGB color space to the Lab color space in order to isolate the L channel, which represents luminance, creating a black and white copy of each colored image. Finally, both colored and grayscale images are normalized to the range $[-1, 1]$.
- **cGAN objective:** As introduced previously, to achieve the recolorization task, a conditional Generative Adversarial Network (GAN) is implemented. The basic GAN structure consists of two neural networks: a generator and a discriminator that are trained in opposition to each other. The goal of the generator is to produce fake samples that are indistinguishable from real data samples, while the goal of the discriminator is to accurately identify whether a sample is real or fake. During training, the generator continually improves its ability to produce realistic samples, while the discriminator continually improves its ability to identify fake samples. This creates a dynamic equilibrium where the generator and discriminator are constantly pushing each other to become more sophisticated. In general, a Generator model can be represented by a map G that take in input a random vector z and outputs an image y: $G : z \rightarrow y$. In contrast, a conditional generative model takes as input both z and an observed image x: $G : \{x, z\} \rightarrow y$. Unlike an unconditioned model, the generator in a cGAN also receives observed data as input, which is suitable for our purpose since we want the model to return a specific colored image based on a given black and white image, rather than generating random colored images. The general loss equation of a cGAN model is the following:

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

However, it has been shown that adding a classical loss to L_{cGAN} does not interfere with the behavior of the discriminator, but instead affects the loss of the generator, whose aim is to produce images that are not only able to fool the discriminator, but also close to the ground truth. To control blurring, an $L1$ loss is chosen over an $L2$, considering also a learning parameter λ, resulting in the total loss:

$$L_{cGAN,tot}(G, D) = L_{cGAN}(G, D) + \lambda \mathop{\mathbb{E}}_{x,y,z}[\|y - G(x, z)\|_1] \tag{2}$$

In previous studies, including the one from Isola et al. [2], it was observed that the model, during training, learned to simply ignore the noise added to the input images. This could result in a suboptimal performance of the model in terms of generating high-quality images. In our study, we address this issue by setting $z = 0$, meaning that no noise is added to the input images during training. This allows the model to learn a more meaningful representation of the true data distribution and generate high-quality images without the influence of added noise.
In practice, during the training of the model, the Binary Cross-Entropy (BCE) loss has been used. The BCE loss is defined as:

$$BCE(y, y*) = \frac{1}{m}\sum_{i=1}^{m}[y_i \times \log y*_i + (1 - y_i) \times \log(1 - y*_i)] \tag{3}$$

For the discriminator, which takes both the real image y and the fake image produced by the generator $G(x, z = 0)$, the BCE loss is calculated for both inputs. When $y* = y$, the true input colored image, the BCE is calculated by comparing it to a vector of $y = 1$ which represents the desired label that the discriminator should be able to assign to a true image. When $y* = G(x, z = 0)$, the desired output should be zero, therefore the BCE loss on the fake input is calculated using $y = 0$. The total loss of the generator is calculated as the sum of the loss for fake and true colored image divided by two. Regarding the Generator loss, the BCE loss is used with $y = 1$ because it represents the label that the generator assigns to its fake image to fool the discriminator. As previously explained, this term is also combined with the $L1$ loss.
- **Generator:** The Generator model is based on the U-Net architecture, which is an improvement from the classi-

cal Encoder-Decoder architecture. Unlike the Encoder-Decoder architecture, the U-Net concatenates the downsampled feature maps with the upsampled feature maps, allowing for a precise localization of the object in the image. The downsampling path captures context information and the upsampling path enables precise localization. This is important in our task as it allows the model to learn the relationship between the black and white image and the corresponding colored image, while also being able to generate fine-grained details in the generated image. A simple representation of its action is represented in 1. Regarding the activation function used in the last layer, the hyperbolic tangent was chosen:

$$\tanh x = \frac{\exp x - \exp(-x)}{\exp x + \exp(-x)}, \tag{4}$$

This function maps its input to values in the range of $[-1, 1]$, which is the same range as the normalized input images. This helps ensure that the predictions of the model are in line with the input values.

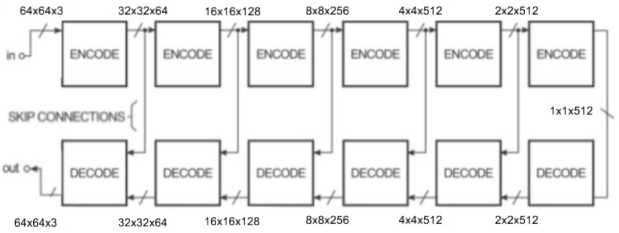


Fig. 1: U-Net architecture used as Generator in the cGAN model. In comparison to the study from Isola et al. [2] the structure we adopted was adapted to take in input grayscale images of shape $64 \times 64 \times 3$.

- **Discriminator:** The Discriminator in a GAN architecture is a crucial component that evaluates the realism of the generated image. The architecture chosen is a PatchGAN, a type of CNN. Unlike CNNs which are trained to classify full images, PatchGANs focus on classifying only small patches of the image at a time. The model takes both the output of the generator (considered "fake") and the "real" colored image as input. The output represents the believability of these $N \times N$ patches of the input images. By running convolutions on the patches across the entire image, the final output of the Discriminator is generated. The choice of using a sigmoid activation function in the last layer of the Discriminator network is important because it allows the network to output a probability:

$$\sigma(x) = \frac{1}{1 + \exp -x} \tag{5}$$

In fact, it is defined in the $[0, 1]$ range, which is also the probability domain. As demonstrated by Isola et al. [2], using $N \times N$ patches with size $1 < N < D$, where D is the image size, is a useful approach to achieve a trade-off between sharp image results with high color diversity (low N values) and softer edges with less color variability (high N values). Unlike Isola et al. who used a 70×70 patchGAN for 256×256 input images, we have built a similar structure with a 46×46 patchGAN for 64×64 images, as the original structure would have produced a patch larger than the image itself. A scheme of the model used can be found in Fig. 2.

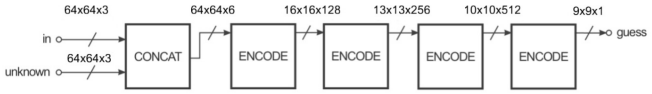


Fig. 2: The $46x46$ patchGAN implemented for $64x64x3$ input images.

- **cGAN with Wasserstein loss:** In order to improve the performance of the model, the loss function used during training was changed in the second part of our study. In general, in GANs models, we consider two probability distributions, P_r which is the distribution of real images and P_g which is the probability computed by the generator. The goal is to minimize the distance between these two distributions in order to generate more realistic images. In generic GANs, this is done by minimizing the Jensen-Shannon distance, which measures the difference between the probabilities associated with each corresponding random variable from the two different distributions. Visually, we demand that the two distributions have the most possible similar vertical shape, but nothing ensures that they are based on the same domain. The main idea behind WGANs is to continuously move P_g towards P_r. Therefore, as a first step, we need a clear definition of the distance between the two distributions, which is defined as:

$$W(P_r, P_g) = \max_{||f|| < 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)]; \tag{6}$$

in [3], Arjovsky et al. define the Wasserstein's distance, where the first term is the mean value over x drawn from a real distribution (a real image) of f, which represents the discriminator function. Now, the latter is referred to as the critic rather than the discriminator because its aim is not to classify patches as done in cGANs. Technically, the last layer's sigmoid function has been removed to eliminate the output range restriction between 0 and 1. The new output will now return high values for far away distributions and low values for close distributions. The second term represents the mean value of the critic's output on fake data (generated images) drawn from x.
The goal of the discriminator is to maximize the separation between these two terms, while the generator aims to minimize their difference. The use of the Wasserstein distance results in a more stable training process compared to traditional GANs.

IV. Signals and Features

The images from Tiny ImageNet have been transformed from the RGB color space to the Lab color space before being input into the model architecture. Unlike RGB color space, which represents images through the addition of three colors, Lab color space includes a dedicated channel called "Luminance" to represent the brightness of the image, with the color information fully encoded in the remaining two channels. This prevents sudden variations in both color and brightness from small perturbations in intensity values that are experienced in RGB color space. Grayscale input images are created by isolating the Luminance channel from a colored image, and this channel is duplicated twice in order to preserve the original image structure of $64 \times 64 \times 3$. This results in input and output images of the generator (gray and colored, respectively) having the same dimensions.

All images have been normalized to the range of [-1,1], considering that color channels in Lab space can take values in the range of [-110, 110], while the Luminance channel can take values in the range of [0,100].

The described preprocessing was performed using a custom PyTorch ImageFolder that returns two $3\times64\times64$ tensors upon input of JPEG images: x tensor representing the grayscale image to be fed into the generator, and y, a tensor representing the true colorization of x, which will be input into the discriminator.

The training and testing sets were created from the "train" and "val" folders provided by Tiny ImageNet. The training set consists of 500 images per class for a total of 200 classes, while the testing set consists of 50 images per class. The Pytorch custom ImageFolder and Dataloader were used to create these sets. The batch size was set to 128 and the data in the training set was shuffled.

V. Learning Framework

Here we proceed with a tecnical description of the implemented Networks and their main differences with previous models.

Generator architecture As stated earlier, the generator is an implementation of a U-Net based architecture, which consists of a downsampling path (encoder) and an upsampling path (decoder).

The downsampling path consists of several blocks, each of which includes a 2D convolutional layer (Conv2d), a batch normalization layer (BatchNorm2d), and an activation layer (LeakyReLU). The first block takes as input an image with 3 input channels, where these three channels represent the same grayscale image repeated three times and obtained as the L channel in the Lab color space of the original image. The output of the first block is an image with 64 feature maps. The subsequent blocks double the number of feature maps while halving the spatial dimensions of the image. The final block of the downsampling path has a spatial size of 2x2 and 512 feature maps.

The upsampling path begins with a 2D convolutional layer that reduces the spatial size of the feature maps to 1x1. The subsequent blocks then successively upsample the feature maps to their original size. Each upsampling block consists of a 2D transposed convolutional layer (ConvTranspose2d), a batch normalization layer, and an activation layer (simple ReLU). The final layer of the upsampling path is a 2D transposed convolutional layer that maps the 512 feature maps back to 3 output channels. The activation function used in this layer is the hyperbolic tangent (Tanh), which produces outputs in the range of [-1, 1]. Additionally, the network has a dropout layer, which randomly sets some outputs to zero during training to improve the network's generalization capabilities and prevent overfitting.

The main difference between the original U-Net model proposed by [1] and this implementation is in the structure of each convolutional block. The downsampling in this implementation is not performed using a max pooling layer, but rather by setting the stride to 2 inside the convolutional layers. This will ideally result in a greater number of convolutional blocks to reach the same dimensionality for 256x256 images, but in the case of the Tiny ImageNet, the first two and the last two layers were respectively dropped from the generator and discriminator models to fit the size of 64x64 pixels.

PatchGAN architecture: The discriminator model used in this experiment is a PatchGAN architecture, as discussed in section II. The PatchGAN classifies patches of the original image, rather than the entire image. The image size in this experiment is small, at 64×64 pixels, so the patch size, $N\times N$, was chosen not to exceed the image dimensions. A simple architecture was selected, with $N = 46 \times 46$, using a 4×4 kernel size for all layers and varying padding, stride size, and output feature maps. The steps in the PatchGAN architecture are as follows:

i. The true and fake images obtained by the generator are concatenated along the channel dimension to form a $6 \times 64 \times 64$ tensor.
ii. This tensor is then convolved with a kernel using *stride* $= 2$, *padding* $= 1$ and 64 feature maps, resulting in a $64 \times 32 \times 32$ tensor passed through a LeakyReLU activation function with a negative slope of $m = 0.2$.
iii. The second convolutional layer has the same padding and stride values, but uses 128 feature maps and includes a batch normalization layer before the LeakyReLU activation function. The resulting tensor has dimensions of $128 \times 16 \times 16$.
iv. The third and fourth layers both have a 4×4 kernel using *padding* $= 0$, *stride* $= 1$ and use 256 and 512 feature maps respectively. Both layers include a batch normalization layer before the LeakyReLU activation function, producing $256 \times 13 \times 13$ and $512 \times 10 \times 10$ tensors respectively.
v. The final layer uses *padding* $= 1$, *stride* $= 1$ and 1 feature map resulting in a $1\times9\times9$ output passed through a sigmoid activation function.

The pixel values of this 9×9 image, which belong in the range [0,1], represent the believability of the corresponding

section of the input image. Each pixel corresponds to the believability of a 46×46 patch of the input image. This patch size can be reconstructed by performing a backward convolution through all the layers of the model, taking into account that this operation disregards the different padding sizes.

W-GAN architecture: The generator model has not been modified in this second architecture. However, the patch-GAN has been modified by removing the sigmoid activation function, as previously justified in section III, transforming it from a discriminator to a critic. The critic function must be 1-Lipschitz continuous, which is achieved by imposing clipping parameters on the discriminator weights. In this case, the clipping parameters were set to $C = 0.01$, as in [3] emerge that too small parameters takes long computation to converge while too big clipping lead to vanishing gradient. The advantages of this procedure are that now the we have a stopping criteria for convergence: if the generator have trained in a good way, the discriminator will not be able to distinguish between real and fake images so it needs to guess with a probability of 0.5. Further more, the expression in (6) will tend to 0 as the two probabilities are getting closer, so empirically this expression is what has been set as loss for the critic model. Other strategies used to implement WGAN accordingly to [3] are the following:

- discriminator has been trained 10 times more per iteration then the generator model
- Adam optimizers has been substituted by RMSprop.

VI. Results

Evaluating the quality of synthesized images is an open and challenging problem. In this report, we made a trade-off between our goal and the available resources in terms of time and computational power. Both networks, the one with the patch GAN and the one with the WGAN implementation, were trained for a total of 100 epochs. On average, the required time per epoch was 5 minutes for the first network and 17 minutes for the second network, using an NVIDIA T4 Tensor Core GPU. The difference in time can be expected as the discriminator model was trained 10 times more than the generator model in the WGAN network. In the following, we will analyze the results from the two networks separately.

cGAN with patchGAN implementation:
In this network, the generative and discriminative models concurrently compete with each other until convergence to an equilibrium is achieved. This behavior is demonstrated by their respective loss functions, as seen in Fig. 3. The discriminator's loss stabilizes at a value of 0.5 from the beginning, which indicates that the generator has fooled it from the start. Meanwhile, the generator's loss starts near 7 and gradually decreases to 4. These values for the generator's loss are not surprising since there is an additional L1 term with a weight of $\Lambda = 100$ to consider. It is also interesting to notice that despite the discriminator do not imporve its performances, the generator is still having a descending trend meaning that the model may benefit from further training.

It is observed that the discriminator can be easily fooled, as seen when comparing our results with those of similar structures trained in [2]. This can be attributed to the reduced dataset used in our training. Despite its large variation in categories, the amount of information it contains is also reduced, which may have compromised the discriminator's ability to distinguish between real and fake images.

This behavior can also be physically seen in the images generated at different epochs, as shown in Fig. 4. This batch of images was chosen randomly from among all the well-predicted test images. It can be observed that the model, after only 25 epochs (column (d)), is already able to produce a good color prediction. It's also interesting to note that the WGAN model, trained for 100 epochs on the same batch of images, fails to improve the quality of the predictions (column (e)).

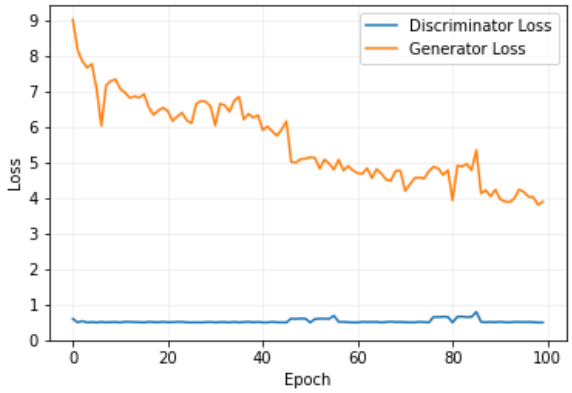
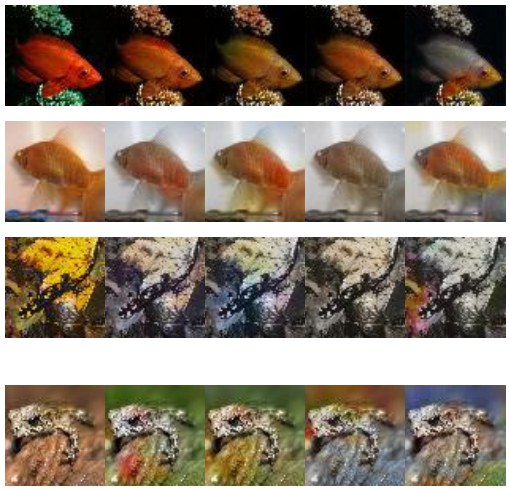


Fig. 3: Loss values for different epochs of training for patch-GAN model

A batch of incorrect predictions is also shown in Fig. 5. The limitations of this model are highlighted here, as it is observed that the base model is still unable to accurately reproduce human skin color after 100 epochs of training. Additionally, it fails to predict colors for unsaturated images with low color variability. This behavior may be a result of the L1 loss added to the generator. It can be noted that in cases of incorrect predictions, the WGAN performs slightly better, as where the base model is unable to depart from grayscale, the WGAN is capable of predicting some color.

WGAN implementation:
In this case the behavior of the losses is very different from the previous ones, this is due to the definition itself of the used metric. For the motivation mentioned in section VI, the loss of the discriminator is expected to approach to 0 while there is no limit for the expectation value that represent the loss of the generator. In Fig. 6 we can appreciate experimentally that even from the very beginning the discriminator loss tends to zero reaching values of 10^{-1} order, the generator loss tends to stabilize itself among a value of 0.6, having a very



(a) (b) (c) (d) (e)

Fig. 4: Good predictions results from the patchGAN model on the test set, according to the epoch of training. Column (a) refers to the ground truth (true label), column (b) represent the results after 100 epochs, column (c) after 50 epochs, column (d) after 25 epochs, while the last (e) offers a comparsion of the same image predicted by the WGAN model trained for 100 epochs.

(a) (b) (c)

Fig. 5: Wrong predictions results from the patchGAN model on the test set, for 100 epochs of training. Column (a) refers to the ground truth (true label), column (b) represent the results after 100 epochs, while the column (c) offers a comparison of the same image predicted by the WGAN model trained for 100 epochs.

slow descending where Arjovsky, et al. seems to obtain more realistic images but with a major time of training.

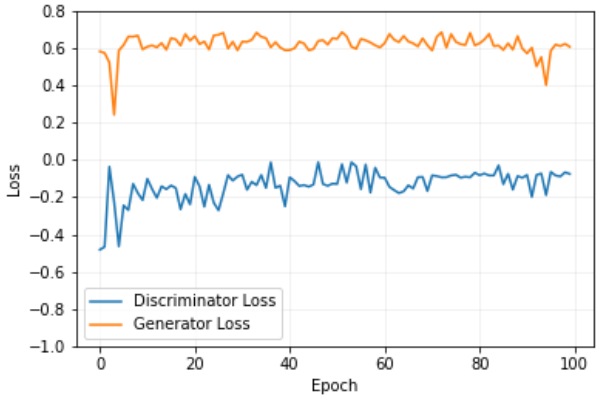


Fig. 6: Loss values for different epochs of training for WGAN model.

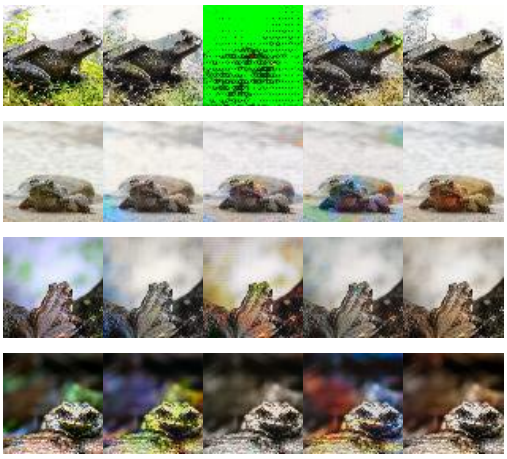
Observing the colorized images in Figures 7, we can state that the WGAN avoids the so-called ”mode collapse phenomenon”. This phenomenon, in the specific case of image colorization, refers to producing a limited number of unique colors that result in sepia or unsaturated tones. However, it can be seen that the WGAN explores a wider range of colors, but in most cases, these colors appear to compromise the realism of the images.

In Figure 7, it is also interesting to note that the improvement in image colorization during the training epochs is not linear. In the first row, a very poor prediction can be observed for 50 epochs of training, while for 100 and 25 epochs of training, the colored image is more realistic. Additionally, patches of color can be observed in most of the images. By comparing columns (b), (c), and (d) with column (e), which represents the same images predicted by the patchGAN base model, it is possible to see that the WGAN model has effectively improved the color saturation of the images and reduced the sepia tone.

In Fig. 8 some of the bad predictions of the WGAN model are presented. It can be observed once again that the model favors unrealistic colorization over unsaturated images. This fact can be clearly seen by comparing column (b) where the WGAN model predictions are presented, with column (c) where the same test images are predicted by the patchGAN model. Both models fail in predicting the colors of these kinds of images, with very different results: noisy and hyper-saturated predictions for the WGAN and gray-scale or sepia-toned images for the patchGAN.

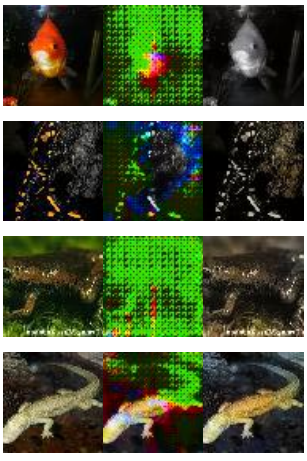
VII. Concluding Remarks

In conclusion, the present report has demonstrated the development of a full Generative Adversarial Network model capable of generating plausible color images from gray-scale images, meeting the expectations and available resources. A comparison between this model and a Wasserstein GAN



(a) (b) (c) (d) (e)

Fig. 7: Good predictions results from the WGAN model on the test set, according to the epoch of training. Column (a) refers to the ground truth (true label), column (b) represent the results after 100 epochs, column (c) after 50 epochs, column (d) after 25 epochs, while the last (e) offers a comparsion of the same image predicted by the base patchGAN model trained for 100 epochs.



(a) (b) (c)

Fig. 8: Wrong predictions results from the WGAN model on the test set, for 100 epochs of training. Column (a) refers to the ground truth (true label), column (b) represent the results after 100 epochs, while the column (c) offers a comparison of the same image predicted by the base patchGAN model trained for 100 epochs.

has shown an improvement in the range of predicted colors, but no significant improvement in the overall quality of the images. Additionally, one of the key benefits of this algorithm compared to previous ones is its simplicity, making it easy to train even on less powerful computing devices.

For future work, it could be interesting to explore different patch-sizes for the discriminator model and to adjust different hyperparameters such as weight-clipping and the number of iterations in the critic model. Furthermore, incorporating a pre-trained image recognition model into the WGAN structure may help overcome the issue of semantic meaning and improve the overall performance.

VIII. Appendix: report project

In this project we have acquired a more robust knowledge on how to implement a Neural Network with Pytorch, how to develope our own customized dataloader and preprocessing functions. We concretize the difference between simple GANs, patch-GANs and WGANs, as well as the difference between classical autoencoders and UNet autocenoders and how to adapt them for different input images. Moreover, we have faced the challenge of writing and organizing a scientific paper. The main difficulties encountered concern the power of our computational resources: compared to more high level pre-trained neural networks our model still lead to poor results, so the challenging part has been trying to evaluate our performances without a similar setting to compare with.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[2] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

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

[4] Y. Le and X. S. Yang, "Tiny imagenet visual recognition challenge," 2015.