Image recoloring with conditional GANs

Alejandro Cano Caldero†, Jesús Moncada Ramírez†

***Abstract*—We present a method so as to tackle the image recoloring problem, that is to say, assigning RGB values to grayscale pixels. Not only will this paper show how this problem can be handled by implementing a UNET Autoencoder and Patch GAN, but it will also showcase an important enhancement provided by additional perceptual losses and the implementation of a W-GAN network with *gradient penalty*. These tools will prove to match human perception well and give higher stability to the training model, respectively. To visualize the progress that has been made, we will gradually attach the refinements previously mentioned and the given results to show how the final implementation outperforms the rest.**

I. Introduction

Image recoloring is already a thoroughly studied field in Neural Networks. However, on this paper we propose a methodology based on additional loss penalties and conditional GANs with little image preprocessing techniques that enhance the performance of the training process.

The use of a U-NET generator and a PatchGAN discriminator is not an unprecedented solution to the recoloring problem. As shown on the paper "Image-to-Image Translation with Conditional Adversarial Networks" (please see [1] for additional information), this approach has proved to give promising results. However, as mentioned on such paper "The encoder-decoder is unable to learn to generate realistic images in our experiments". Given this situation, our method endeavors to match human perception by resorting to additional perceptual losses and variations in the conditional GANs architures.

One of the main problems in image recoloring is to generate realistic images, that is, to assign appropriate RGB values to each pixel so as to output a coherent image from a human's point of view, while maintaining a stable training. To tackle the given problem, this paper proposes the implementation, on a toy dataset, of *The Learned Perceptual Image Patch Similarity* (*LPIPS*) in the generator's architecture and the use of a *WGAN* architecture as the discriminator with *gradient penalty* (please see [2] for further information). This paper intends to show the potential of these techniques when trying to generate more realistic images in image-to-image tasks.

In a nutshell, our contributions are:

- On toy datasets, we demonstrate that realistic results can be achieved with *LPIPS*.
- It is also proposed the use of *gradient penalty* to overcome the issues of standard GANs regarding training stability.
- It is shown the improvement over the original U-NET and PatchGAN architecture.

The structure of this report goes as follows. In Section II, we describe some related work and contributions that have made this project possible. Subsequently, Section III and Section IV will describe the high-level given solution and signal processing techniques, respectively. The learning strategy will be presented in Section V. Finally, we will show the final result in Section VI and some concluding remarks in Section VII.

II. Related Work

Conditional Adversarial Networks. The use of a U-NET and PatchGAN architectures has been proved to be really effective for image-to-image transformations [1]. Further information about the effectiveness and other applications of both the U-NET and PatchGAN architectures can be found in [3] and [4], respectively. Several other papers have used deep convolutional neural networks for image colorization. For instance, in this paper [5], very artistically impressive results have been achieved. However, turns out that in those results, none of the generated colors actually matched the corresponding real-life scene, resulting is less realistic images from a human's point of view. There are other approaches based on color distributions associated with objects [6] where they show compelling results. Yet, they represent work in progress and they are not focused on the generation of realistic color transactions.

Our work differs from the previously mentioned papers in some perceptual losses and penalties. Unlike former works, we use *The Learned Perceptual Image Patch Similarity* [7] which introduces human perceptual similarity judgments to generate more realistic images.

Our method also uses another approach for the discriminator architecture. We introduce *gradient penalty* for the discriminator loss. This method has been proved to outperform standard WGAN (proposed in [8]) and enables stable training [2].

III. Processing Pipeline

In our experiments we have included 4 architectures related to the proposed problem. First, we implemented the same structure shown in [1] and, after that, we implement *LPIPS*

†Department of Information Engineering, University of Padova

and *Gradient Penalty* **separately**. Finally, we combine both improvements so as to train our final model. The structure of our approach goes as follows (a brief and visual representation is depicted in 1):

- **Image processing**. In the first place, after loading the data, not only do we normalize the values for each pixel of each image, but we also resize them. Moreover, some other advanced pre-processing techniques have been used (Gaussian blur and histogram equalization).

- **Dataset implementation**. As we will nurture the discriminator with pairs of images and we need to compare input and target images, we decided to define a class that will be in charge of, given both the colored and gray-scale data (already preprocessed as we mentioned in the former paragraph), returning tuples of images, that is to say, a gray-scale image and its corresponding colored/target image.

- **Generator and discriminator**. We use a U-NET architecture for the generator and a PatchGAN architecture for the discriminator, as shown in [1]. The generator will be provided with a single gray-scale image and the generator with two images (given by the previously mentioned class we defined earlier), either a real image or a fake image (generated image) alongside the same input given to the generator.

- **Loss functions**. On the one hand, the defined generator losses will endeavor to penalize the generator in case it did not succeed in fooling the discriminator (*Binary Cross Entropy*) or when the output image does not match the colored/target image quite well (*L1 and LPIPS*). On the other hand, the discriminator losses will try their best to penalize the discriminator when it cannot distinguish a fake image from a real one (we use *gradient penalty* so as to endow the NN with more training stability.

- **Training**. In our experiments, in order to train our model, we go through the whole dataset (hundreds of images) 10 epochs. During the training process, we store the losses and save the models.

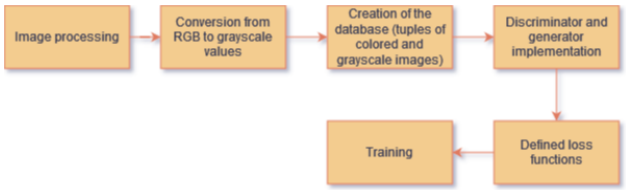

Fig. 1: High-level description of our processing pipeline

IV. Signals and Features

For the pre-processing pipeline, we have resorted to transform modules. We will distinguish between the transformations required by the architectures of the generator, discriminator, and our project itself, and the transformation performed to enhance image quality and facilitate the training process.
In the first step, we use Pillow to read images and then we convert our PIL images into Pytorch tensors. After this, we apply normalization. As the final activation function of the generator is $\tanh$, meaning the output goes from -1 to 1, we scale down the input signal from [0 - 255] to [-1, 1] so as to ensure optimal comparisons over the data. An image resizing process must also be performed. For our experiments, all the images get resized to 256×256 in order to maintain the aspect ratio of every input image. An additional step is needed in order to build our final dataset, as we need to generate gray-scale data from our RGB PIL images.

As for image pre-processing methods to improve image quality, we have selected Gaussian blur and histogram equalization. The Gaussian blur, with $K = 5$ (K being the kernel size) and $\sigma = 0.7$ removes Gaussian noise that may be present. The histogram equalization increases the image contrast, which is especially important in grayscale images, as we discovered they looked extremely dark. Both methods will facilitate the training process in order to achieve better results.

In the first version of the project, the whole ImageNet dataset was going to be used. After realizing our hardware limitations, we chose to use Imagenette (please see: https://github.com/fastai/imagenette for additional information), a reduced version of the first one. Finally we were forced to include only 500 samples of Imagenette in our training set; our test set is made up of 150 samples (30% of training set).

V. Learning Framework

A. Generator

A U-NET architecture has proved to be impressively effective for image-to-image problems like image recoloring. In our work, we implement this type of network with modules of the form *convolution-BatchNorm-LeakyReLu* and a normal distribution for weight initialization. As proposed in [1], a Gaussian distribution with $\mu = 0$ and $\sigma = 0.2$ gives promising results. Our generator will perform one gradient descent step at every 5 gradient descent steps of the discriminator, that is, we alternate between a step of the generator and 5 other steps of the discriminator. This differs from the implementation given in [1], as we implement a WGAN structure [8]. So as to optimize our implementation, we use SGD with minibatches of size 1 and the Adam's optimizer (for our experiments we use the values 0.0002 and 0.5 for both the learning rate and weight decay, respectively). As for the generator losses, we implement *Binary Cross Entropy* alongside the *L1* (captures the low frequencies) with

a scaling factor of 100 and *LPIPS* with a scaling factor of 50. For the last parameter, in our experiments, we have used values from 0 to 50, but 50 turned out to give the best results. A visual representation is shown in 2.

B. Discriminator

It is known that *L1* enforces low-frequency correctness. Given such property, in our work we use a *PatchGAN* architecture to focus the attention only on local image patches (as shown in [1]). Unlike previous papers, we use a WGAN architecture with modules of the form *convolution-BatchNorm-LeakyReLu*, but it is important to highlight that we also implement *gradient penalty* since it has been shown that that technique performs better than standard WGANs [2]. Likewise, for weight initialization we use a Gaussian distribution with $\mu = 0$ and $\sigma = 0.2$. Regarding the discriminator losses, it has the real images output predictions ground-truth label as 1 and the fake images output predictions ground-truth label as 0. We compute the *original critic loss* (the labels are substracted from the output predictions) and then multiplied by the *gradient penalty* with a scaling factor $\lambda = 10$ (as shown in [2]). Our new objective is depicted in equation 1. As for optimization techniques, we use SGD with minibatches of size 1 and the Adam's optimizer (we implement the same parameters used by the discriminator). The diagram describing such architecture is 3.

$$L = \underbrace{\mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_g}[D(x)]}_{\text{Original critic loss}} + \underbrace{\lambda\mathbb{E}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}[(\|\nabla_{\hat{\boldsymbol{x}}}D(\hat{\boldsymbol{x}})\|_2-1)^2]}_{\text{Our gradient penalty}}. \tag{1}$$

C. Training

For the training process we define 10 epochs over a toy dataset comprised of 500 images, that is, 5000 iterations. On each iteration we retrieve a batch comprised of a pair of images (batch size is 1); the input and the target image. Right after that, we zero out gradients and create 30×30 vectors of ground-truth labels (0's & 1's) (the dimensions of these vectors are accounted for by the dimensions of the output prediction from the discriminator). Subsequently, the generator is provided with the input image and outputs a fake one. At this point, the discriminator comes to the fore, and performs 5 (*critic iterations*) gradient descent steps. On each step it takes two pairs of images, i.e, an input comprised of both the same input image provided to the generator alongside the generated image and another input containing both the same input provided to the generator alongside the target image, returning two predictions. Now, it is time to compute the discriminator losses, so we compute the gradient penalty, the losses previously described and we store such losses for further analysis. Finally, we take one gradient descent step in the discriminator. So as to train our generator, we compute a prediction from the discriminator and the generator loss based on such prediction. In the same way, the final step would be to store the losses and take one gradient descent step in our generator.

D. Experiments

We want to devote an additional section to introducing a brief description of the different experiments performed for this paper.

In the first place, we implement the original *pix2pix* described in [1]. Secondly, we only implement *LPIPS* as we decribed earlier and test its performance, that is to say, we do not use the WGAN structure, yet. In the next part of the project, we finally implement our WGAN structure with *gradient penalty*, but on this occasion, we neglect *LPIPS*. Finally, we combine both the additional perceptual losses and WGAN so as to compare its performance with the previous experiments. The results will be shown in the next section VI.

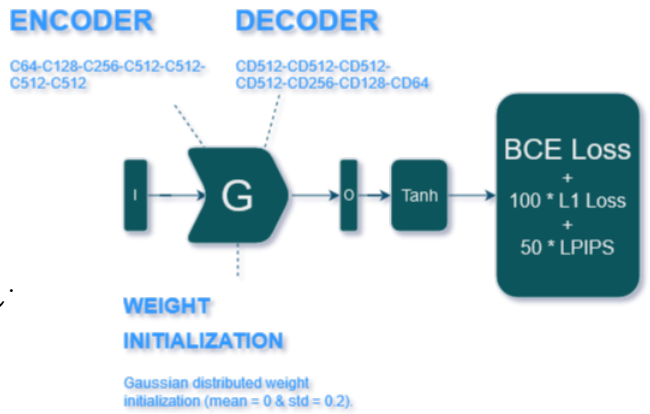

Fig. 2: This is an image of the U-NET generator structure with detailed information about its implementation.

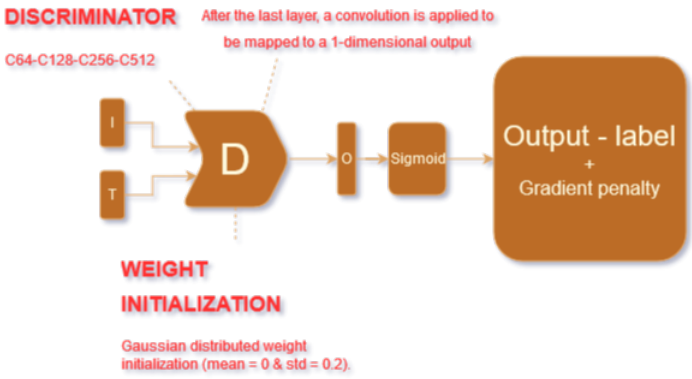

Fig. 3: This is an image of the WGAN structure with detailed information about its implementation.

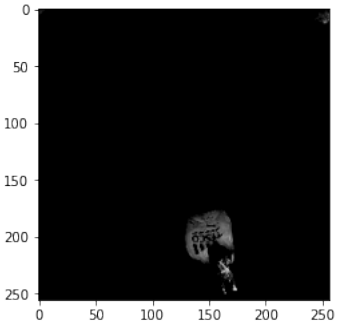

Fig. 4: Input image 1

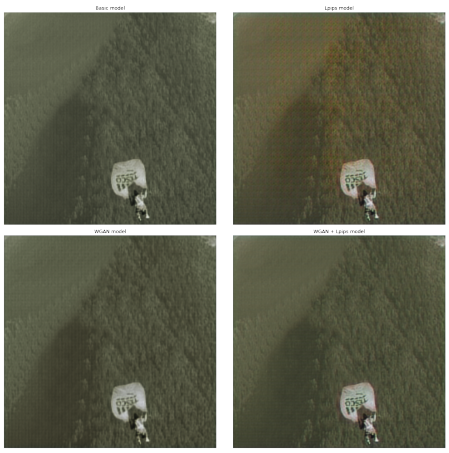

Fig. 5: On the upper left corner, we can see the final result when using the original implementation. On the upper right corner, the image corresponds to the LPIPS implementation. The picture on the bottom left corner belongs to the WGAN architecture. The last image is our final result when combining both LPIPS + WGAN

VI. Results

A. Empirical results

After the training process we have obtained promising results that meet the requested objective with an observable improvement over the original proposal. However, we can still observe that the final generated images have a sepiaish hue. When doing some research, we have discovered other papers where the same issue happens even with larger training datasets. This does not mean that these results cannot be improved by performing more iterations over a larger dataset. Nevertheless, as we mentioned in the first place, we can already tell the difference and improvement accounted for by the implementation of our additional techniques. In picture 5 we can see the generated images given the input image in picture 4.

We can observe that colors in the image generated by our final implementation are slightly more vibrant and there is higher contrast. All in all, this leads to a better reconstruction of the input image. Note that the generated image from our LPIPS implementation contains small squares scattered over the image, meaning a coarser resolution.

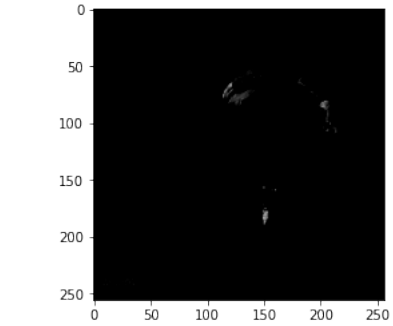

Fig. 6: Input image 2

Fig. 7: The sort of representation is the same we followed in 7

Another example is depicted in pictures 6 7

B. Training curves

It is well known that the discriminator and the generator losses must converge, meaning that the generator loss will gradually decrease 8, that is, it is gradually getting better at creating realistic data, while the discriminator loss will increase over time 9, as the generated images provided to the discriminator are more and more convincing.

C. Analysis of the generalization error

The generalization error is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data. In order to do this, we have resorted to a group of images, based on the previously mentioned data separation in IV, which has not been used to train each model.
The final results vary depending on the trained model. The outcome is depicted in this table 1

As we can see, our estimation of the generalization error of our final implementation does not have the minimum value. Nevertheless, the discriminator error is higher compared to the others. According to our studies, that means we do not get extremely similar images to the input data but it is challenging to distinguish the generated images from the real ones.

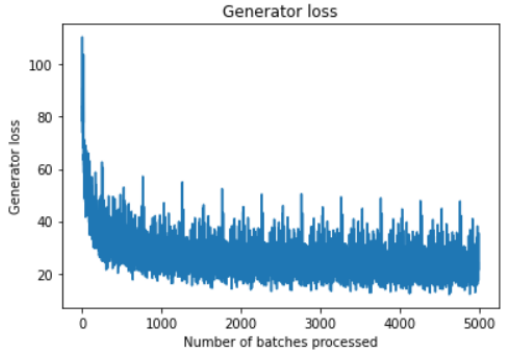

Fig. 8: Generator loss of our final implementation as a function of the number of batches

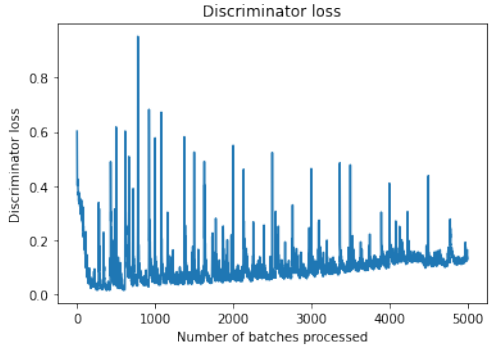

Fig. 9: Discriminator loss of our final implementation as a function of the number of batches

Model	Generator error	Discriminator error
Original	8.9982	0.3435
LPIPS	10.4122	0.4239
WGAN	11.0813	0.6886
WGAN + LPIPS	10.6731	0.8153

TABLE 1: Estimations of the generalization errors

VII. Concluding Remarks

In a nutshell, the final results shown in this paper are accounted for by an application of a U-NET and PatchGAN architecture with additional *perceptual losses* and *gradient penalty* on a toy dataset. Such results have shown that there is still a long way to go, specially when it comes to obtaining more vibrant colors. Nevertheless, we can clearly contend that the improvement over the original architecture is significantly notable.

This project has shed some light on all the problems and issues that everybody has to face when dealing with Neural Networks. Putting aside all the technical aspects taught in class, we have learned how to do research on other projects and papers provided by the community and the key points when writing a compelling paper. We have had some difficulties related to hardware limitations, as we have not succeeded in training our models as much as we wanted. To be honest, we have encountered also many difficulties mainly because we were not acquainted with all the necessary metrics and assessments that should be performed to properly address these projects. In fact, we are aware of the fact that some sections/parts of this project can be further improved.

VIII. Contributions

A. *Cano Caldero, Alejandro*

Alejandro has been in charge of the **implementation** of the two basic classes of our project, that is, **the generator and discriminator**. Also, he has been devoted to creating the different **loss functions** and implementing the **LPIPS and WGAN variations**.

B. *Moncada Ramírez, Jesús*

Jesús has been in charge of the **creation of the dataset** and the implementation of the **training loops** for each model. He has also created several **useful functions** so as to deal with the models and others to compute some measures like the **test error**.

As for the **paper**, both collaborators have **equally contributed**. Actually, **we believe that we both have equally contributed to the elaboration of the overall project.**

References

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image translation with Conditional Adversarial Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] I. Gulrajani, F. Ahmed1, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," in *Advances in Neural Information Processing Systems*, Dec. 2017.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, May 2015.

[4] C. Li and M. Wand, "Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, Apr. 2016.

[5] J. Hwang and Y. Zhou, *Image Colorization with Deep Convolutional Neural Networks*, 2017.

[6] M. Afifi, B. Price, S. Cohen, , and M. S. Brown, *Image Recoloring Based on Object Color Distributions*, 2019.

[7] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jan. 2018.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Jan. 2017.