Image Colorization using Cycle-GAN

Md Mehrab Tanjim, and Jixuan Liu
Computer Science & Engineering
University of California, San Diego
California, CA 92037
{*mtanjim*, *jil*046}@ucsd.edu

Abstract

Automatic colorization of grayscale images has been a hot topic in computer vision research community. Not only it is an interesting and fun task (e.g. bringing life to black and white images) but also a very practical and important one (e.g. colorizing secutity camera footage). In advancement of CNN and Generative Adversarial Network, image colorization task has been approached in both supervised and unsupervised way. In this paper, we seek more ways to establish true color by applying Cycle-Consistent Generative Adversarial Networks (Cycle-GAN). To the best of our knowledge, Cycle-GAN has not been applied to image colorization task before. Thus our work not only presents a new work in this area but also explores some of the opportunities in research direction.

1 Introduction

Image colorization is a task which maps each pixel value from a grayscale image to three (in case of RGB image) or two (in case of LAB image) pixel values. At first this task can seem very challenging due to the fact that we have only one dimensional information for a multidimensional data. In reality, children face this challenging problem every time they color a black and white image. And they solve it with minimal effort also, because we as a human, are capable of linking semantic cues to image. For example, we know at daylight, the shades of buildings are different, and so, if we are asked to color the background, we will normally color with skyblue. While this kind of colorization scheme works for most of the objects, it certainly is not a generalized approach. For example, if there are different types of balls, and they have different colors, it may not be easy to assign color based on object. However in image colorization, our goal is not necessarily to match the ground truth color, but a good enough color that will fool a skilled human observer. Thus to produce visually indistinguishable results, the task is to capture enough of the statistical dependencies between the context and the textures of grayscale images and their color schemes.

Beside simply fascinating from an aesthetics point of view, image restoration task has wide application in real world. If we have an automatic way to coloring images, we can apply this technique, in restoring black and white surveillance video to colored one, we can enhance and change color of an image for better inseparability. That is why automatic colorization of black and white images has been subject to many research within computer vision and machine learning communities.

This task has been approached in different ways in the past. Before the advancement of CNN techniques, most focus was to build a statistical model for coloring the images. But in recent year, CNN model has emerged as state-of-the-art technique for image classification problem, achieving as low as 4% error. Their breakthrough success can be mostly attributed to their ability to learn and discern colors, patterns, and shapes within images and associate

them with object classes. Thus, researchers in the past have tried to apply this technique to generate realistic colorful images.

But CNN models have a major weakness in a task like image colorization. CNN models are capable of learning hidden representation of objects which help them correctly identify an object, but it does not inherently allow it learn the underlying distribution, which is very important when it comes to generate colors. Recently generative adversarial networks [2] have been proposed to generate vivid images from some random noise. It is composed of two parts: a generative model G, and an discriminating model D. Generative model G tried to map from random noise to original distribution and generate fakes, and discriminator tries to catch the fake one from reals. This adversary goes back and forth, and after some time, they will come to an equilibrium. By conditioning on the noise, we can generate realistic images. That is called conditional GAN[1].

Recently, another kind of GAN has been introduced which is called cycle-consistent GAN or cycle-gan in short [6]. It has shown a lot of promise in generating picture of different domain. But to the best of our knowledge cycle-gan has not been used in a task like image colorization. In this paper, our goal is therefore, to explore various opportunities of cycle-gan in image colorization task. We work with two datasets: one showing the weakness of vanilla CNN models, and another for conditional GAN. In comparsion, we show cycle-gan is capable of generating more realistic images in both cases.

2 Related Work

As mentioned earlier, image colorization task has been approached in both hand-crafted and automated way. In non-parametric methods, given an input grayscale image, the task is to define one or more color reference images (provided by a user or retrieved automatically) to be used as source data. Then, color is transferred to input image following the Image Analogies framework [10]. There is also parametric methods which treats this as a regression problem and makes prediction from large dataset [11]. The automatic way is applying CNN architecture. In [9], they used additionally trained model to handle diverse colorization of a scene. Further, there have been work which posed this problem as a classification problem for using CNN model [12]. As image colorization is generation task, and autoencoders can act as a generator, researchers have used variational encoder to learn the low dimensional embedding of color fields [13]. As far as applying GAN for image colorization is concerned, there have been some work which includes conditional GAN [1], but no work has been done with cycle-gan.

3 Motivations

As we mentioned in the introduction, image colorization task is task of generation, for which vanilla CNN models are inadequate. As GANs are more suitable for learning distribution, we have chosen to apply GAN in restoring colors. For our choice of GAN model, we have chosen cycle-gan because of couple of resasons. First, it has shown promising results where we do not even have a paired set of images. So, our hope is that for paired task, cycle-gan should provide better results as well. Second, the opportunities and challenges of applying cycle-gan in image-colorization task has not been explored before. Therefore, in our work, we try to go through some of the issues we faced when we applied cycle-gan, and share some of our insights how cycle-gan can be used in generating vivid colors in difficult datasets.

4 Background

4.1 Vanilla CNN Models

Convolutional neural networks(CNN) are usually used in tasks like images classification, objective detection, face recognition etc. CNN is usually composed of different layers included convolutional layers, pooling layers and fully connected layers. It allows the error to backpropagate through the network to update the parameters. As the neural networks

are going to deeper and deeper to get good performance in these years, it is used in many other fields. Here, we used vanilla CNN model for image colorization.

4.2 Generative Adversarial Network

Generative adversarial networks (GANs) was first proposed by Ian Goodfellow[2]. Since then it has attained much attention in unsupervised learning. The basic way it works as follows: there are two separate parts of neural networks: the Generator and the Discriminator. Intuitively, the Generator will try to create some data as real as possible to try to fool the Discriminator. And the function of the Discriminator is to detect the authenticity of the data from the Generator. It will output 0 as fake and 1 as real to evaluate the performance of the Generator.

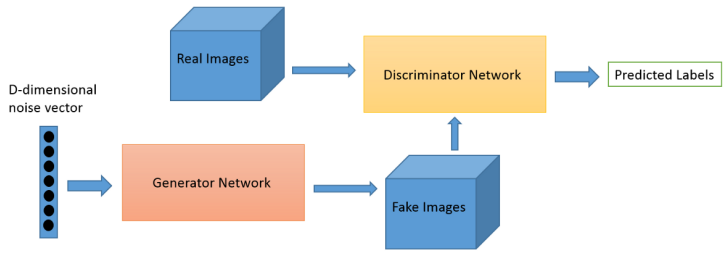
The loss for of the vanilla GAN is:

$$\min_{\theta_g} \max_{\theta_d} \left[E_{x \sim p(data)} \log D_{\theta_d}(x) + E_{z \sim p(z)} \log \left(1 - D_{\theta_d}(z)\right) \right] \tag{1}$$

The θ_g here are the parameter of the generator, and θ_d are the parameters of the discriminator.

The following figure shows how it works.

Figure 1: Generative Adversarial Network, photo credit[5]

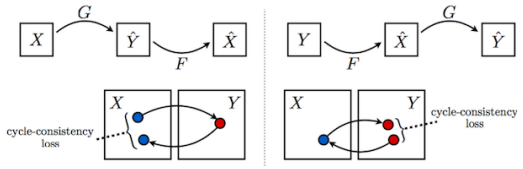


The problem with vanilla GAN is it generates from random noise, thus there have been some works on improving this. Conditional GANs are such an improvement which conditions on original input along with noise. It has been used in many computer vision tasks, like text to image generation[3], image-to-image translation model [4], etc.

4.3 Cycle-consistent Generative and Adversarial Network

[6] introduces cycle-gan which has two additional generators : one maps from X -> Y (G) and another Y -> X (F), and corresponding Adversarial discriminators Dx and Dy, G tries to translate X into outputs, which are fed through Dy to check whether they are real or fake according to Domain Y. Role of F is similar to G, except it is reverse. The real novelty of cycle-gan lies in cyclic consistency loss in addition to the Generator and Discriminator loss (as described above) which is shown in the figure below:

Figure 2: Cyclic Consistency Loss, image credit[6]



This kind of loss uses the intuition that if we translate a sample from Domain X to Y using mapping function G and then map it back to X using function F, how close are we from

arriving at the original sample. Similarly, it calculates the loss incurred by translating a sample from Y to X and then back again to Y. This cyclic loss should be minimized. The cycle consistency loss is[6]:

$$L_{cyc}(G, F, X) = E_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1]$$

Putting the consistency loss and generative gan loss together, the objective of the cycle-gan is[6]:

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F, X) + \lambda L_{cyc}(F, G, Y)$$

Apparently, Cyclic Losses are so important that they are multiplied by a constant lambda (in the paper the value 10 was used). In addition, [6] introduced the identity loss, which is we give the Y to function G instead of X, it should be able to produce Y nonetheless (same applies for function F). This loss is given by following:

$$L_{identity}(G, F) = E_{y \sim p_{data}(y)}[\|G(y) - y\|_1] + E_{x \sim p_{data}(x)}[\|F(y) - x\|_1]$$

when it is added to total loss, it is added with multiplied by an additional hyperparameter which is $\lambda L_{identity}$. If we do not wish to calculate this loss, this hyperparameter can be turned to zero.

5 Experiments

5.1 Dataset

We worked with two datasets. The first dataset is Texas Innovation Challenge dataset: [7] provides LAB space images of MIRFLICKR25k[8] randomly sized colored image dataset. We chose this dataset because it has nice constrast, for which we believe vanilla CNN model will perform poorly. It has 25K images from which we only use 1000 for training and 100 for validation due to limitation of computation resources and duration of training time. We transformed the LAB space to RGB space, and normalized the values.

The second dataset is Oxy-Flower Dataset[14]. We used the full dataset: 1020 images for training, 340 images for validation. We chose this because apparently conditional GAN performs poorly on this dataset [15]. So, it can be a good challenge for cycle-gan. We worked with both LAB and RGB version, and in our experiments we work with different transformation, like normalization, without-normalization, etc.

5.2 Models

5.2.1 Baseline CNN Architecture

For baseline, we implemented a simple 4-layer convolution neural networks and an encoder-decoder CNN architecture (10 layers) to the map the gray images. The 4-layer CNN model has layers without pooling and fully connected layers.We Keep the width and height of the image same in all layers, and increase the channels of the feature maps in the first three layers(16, 32, 64) and decrease it to 3 as the output channel of restored images.

Then, we change this 4-layer CNN to an encoder and decoder architecture. In the first 5 layers, the width and height of the feature map are being decreased (224 to 7), and the number of channels are increasing (3 to 512). And in the last 5 layers, we changed the features size inversely. The output is $224 \times 224 \times 3$ rgb images.

5.2.2 Cycle-gan Architecture

We build the basic architecture of cycle-gan from the authors of cycle-gan[6]. We have applied cycle-gan in the following manner: the generator A takes the color image and produces gray version and the second one does the reverse part. For colorization the generation B network is used. The discriminator A forces the network to generate realistic colorized scan, and discriminator B forces the network to learn realistic B/W scan. This is shown in Figure 3 and Figure 4. We used a resnet for the generators. The resenet is similar to the encoder and

decoder architecture of our 10-layer CNN model except that it now has 9 residual blocks and we are using 3 layers for encoder and 3 for decoder. Encoder downsamples with kernel size 3, stride of 2 and padding of 1 and increases the number of channels from 3 to 256 channels. The upsampling is just the reverse of the downsampling. In between, we have 9 residual blocks for better passing the gradients. We applied instance normalization and relu at the end of each convolution. At the last layer we produce output by Tanh activation. The discriminator is just a bunch of convolution done one after another. We used four convolution layers from 64 channels to 512, with kernel size of 3, stride of 2 and padding of 1. After each convolution we applied instance normalization and leaky relu activation. The last layer is a fully connected layer for prediction.

Figure 3: Cycle A

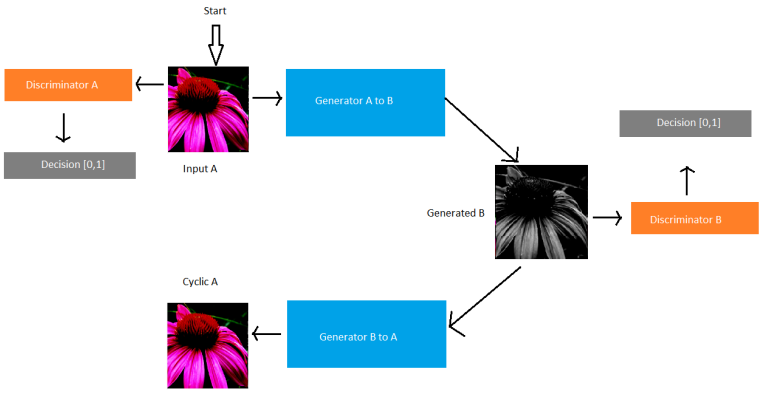
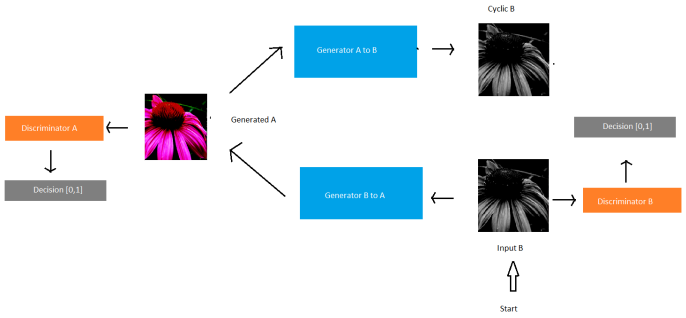


Figure 4: Cycle B

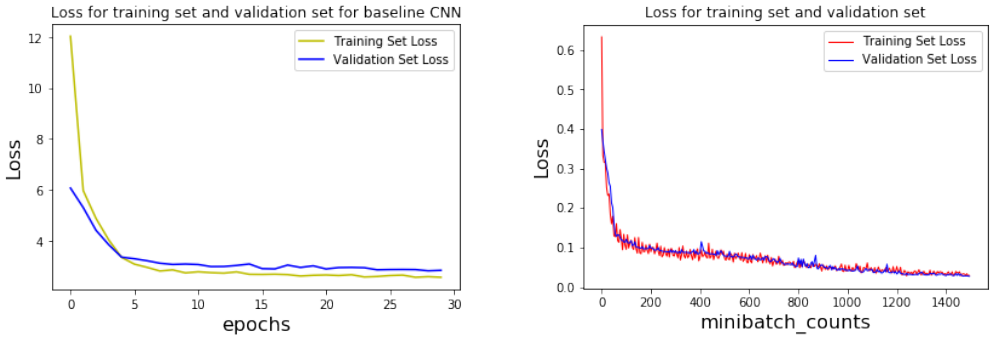


6 Results

First we discuss results from the baseline CNN model for the texas innovation challenge dataset. Figure 6a shows the recovered images from the test set after training and Figure 5a shows the loss. Compared to the gray images and ground truth colorful images, the recovered images may not have the satisfied results, and they all seem to have brown filter. This is to be expected as our CNN model has only 4 layers without much model complexity.

Next, we show the results for the 10-layers CNN model. Figure 6b shows the recovered images from the test set after training and Figure 5b shows the loss. It seems to work better than the 4-layers CNN model. The 10-layers model try to learn the RGB value for different

part of images compared to the simpler CNN model. However, the images output of 10-layers model are pretty blurry. It is because that the model works like encoder and decoder, so it will lose some detail information about of the original gray images and pay more attention on the RGB values. However, the feature map of width and height of the 4 layers-CNN model doesn't change, so it could keep the original edges or some shape detail compared to the 10-layers-CNN model.



(a) Loss of Baseline 4-layers CNN (b) Loss of Baseline 10-layers CNN

Figure 5: loss for training and validation set of 4-layers and 10-layers CNN model

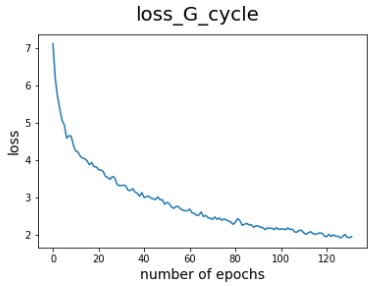
(a) validation output of the 4-layers CNN model (b) validation output of the 10-layers CNN model

Figure 6: Generated Image from 4-layers CNN and 10-layers CNN model, left to right: gray image, ground truth, recovered colorful image

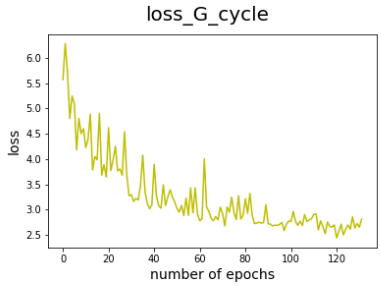
Now we show results for cycle-gan for same dataset. For cycle,as discussed in the background, we have different kind of losses. All the losses are shown in the next Figure 10. The first row presents the cycle-consistency loss for both train and validation. We were able to run 132 epochs with our dataset, and it showed that our model was able to learn the cycle mapping. The second kind of loss is identity loss, which measures the loss if same input is fed to the network. For example if converting from Color to Grey, for identity loss, we have to feed the original Grey image, and see how much it differs. This forces the network to learn internal representation of color maps. The next loss is the GAN loss, which as epoch goes by tries to converge to 1. Finally all these are compiled into single generation loss. The last row shows the loss for discrinimator which is below 1.

Figure 8 shows some of the generated images. From left to right, the first image is the real image, then the real B/W, then generated color image, and then generated B/W. We can certainly judge that images generated by cycle-gan are vivid are realistic in color.

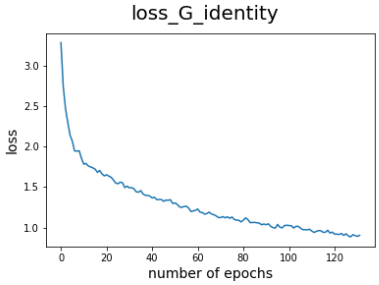
But real interesting thing happens when we applied cycle-gan to oxy-flower dataset. For oxy-flower dataset, [15] provides some results from conditional GAN. The author showed that this image was difficult for conditional GAN to learn because most of the images have green leaves and green background, and that is why there is a generic green filter added to it. Figure 9 shows that this result: This dataset thus provides some challenges even for GAN network, so we think this is good opportunity for exploring cycle-gan's capability. At first we did not make any changes, and applied cycle-gan as it is. It was giving us the same kind of green filers. Figure 10a shows this effect for a validation image. There were some grid



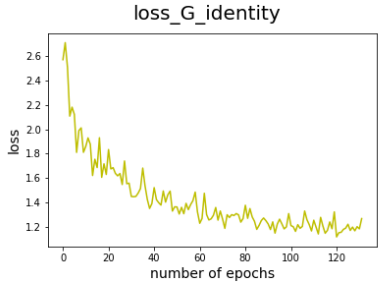
(a) Cycle loss of Generator in train data



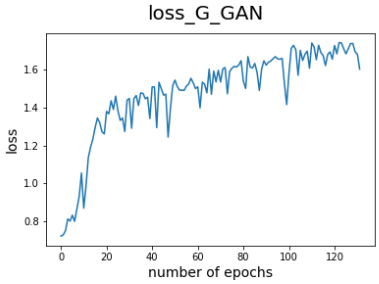
(b) Cycle loss of Generator in validation data



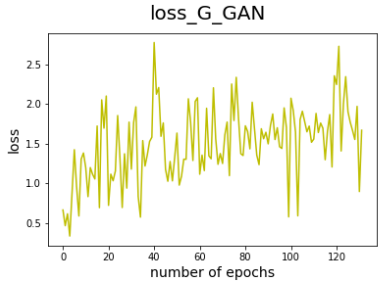
(c) Identity loss of Generator in train data



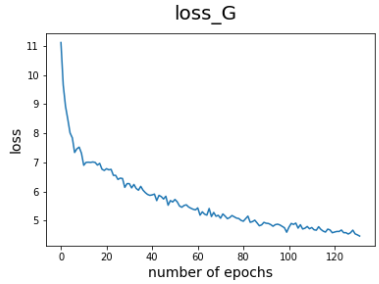
(d) Identity loss of Generator in validation data



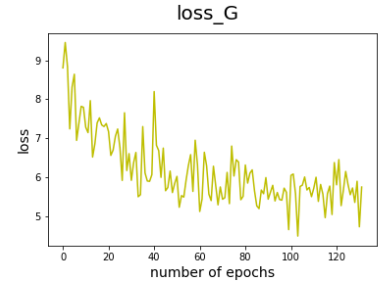
(e) GAN loss of Generator in train data



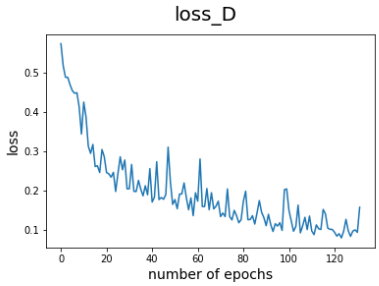
(f) GAN loss of Generator in validation data



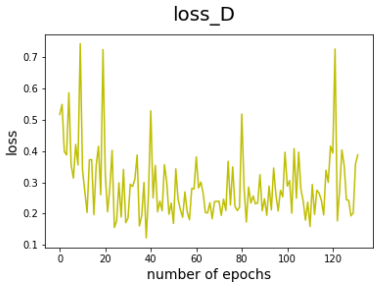
(g) Total loss of Generator in train data



(h) Total loss of Generator in validation data



(i) Total loss of Discriminator in training data



(j) Total loss of Discriminator in validation data

Figure 7: Loss of cycle-gan model

Figure 8: Generated Image from cycle-gan, left to right: Real A (Color Image), Real B (Grey Image), Fake A (Color Image), Fake B (Grey Image)

Figure 9: Generated Image from Conditional GAN for Oxy-flower dataset[15]

effect also, but we suspect it is because of running less number of epochs (100). So far we were applying RGB channel for ground truth. At first we thought, if we experimented with LAB space, the results might get better, because then it only had to learn two more spaces. But figure 10b shows that is not case, and we are seeing a kind of AB filter along with grid in the outputs.

(a) Cycle-GAN applied in RGB space

(b) Cycle-GAN applied in LAB space

Figure 10: Generated Color from Cycle-GAN for Oxy-flower dataset

We also tried to change different type of preprocessing like normalizing, horizontal flip, etc, but it was not working good. So, we thought one of the reasons may be we are using paired datasets and the networks is learning the most common feature in all images, which is the green background and leaves. So, it may be possible if we can train on different types of pairs with shuffling, so that we have gray images of a flower and colored images of a different flower. Shuffling will make sure that the network knows about the variation of colors in images. All these is actually an instance of "unpaired" training but instead of having colorful images from two domains, we actually are training unaligned images of same domain, only difference is one image is black and white and another is colorful. We also change the direction this time, we treated black and white images as Real A, and colorful different image as Real B. After running 25 epochs the results were very promising. Figure 11 shows the result. Losses for this dataset was similar to that of Texas Innovation Challenge and are not shown.

Figure 11: Generated Image from Cycle-GAN, trained in Unpaired Fashion. From left to right: Real A, Real B, Fake A, Fake B

7 Discussion

As mentioned in the introduction, similar looking balls can have different type of colors, but it can be seen in Figure 8 that cycle-gan actually colored this all the same manner. This problem is similar to the problem of oxy-flower dataset, where we were observing a greenish filter. As we saw, this problem can be mitigated if we can train this on similar kind of objects more times or in unaligned manner. In Figure 11, we can see that we can get more colorful flower images. We can also observe that network A when trained on Real A (gray) it generates a gray image but transforms the flower to a different flower but same as Real B. The actual colorization of Real A happens in Fake B which is transformed from Real B. This kind of phenomenon is unseen in cycle-gan. This actually shows the model is actually trying to learn some color schemes. In our opinion this is a promising direction for applying cycle-gan.

8 Future Work

Due to scarcity of computational resources and training time, we could not run on whole datasets, so in future we would like to include all of them. Also, we believe the unpaired training needs more study. So, we would like to do an ablation study, where we freeze one cycle network, turning off identity loss, etc. Also, if possible we would like come up with a performance metric for this task. Distance Based Metrics: RMSE (Root Mean Square Error) and PSNR (Peak Signal-to-Noise Ratio) are inadequate because of nature of the task (i.e. it is not classification task) and currently colorization Turing Test is used where human decide whether the generated color is realistic or not. So, another direction is to come up with a performance metric on which we can optimize.

9 Conclusion

In this project, we have approached the image colorization task in a new way. We applied cycle-gan in image colorization task in both paired and unpaired fashion. We compared our the results with both vanilla CNN and condition GAN, and showed that Cycle-GAN has a lot of promise in the field of colorization. We saw for a challenging dataset like oxy-flower, the unpaired training worked better, and produced some interesting results. We believe further study is needed in this direction and our work is a first step towards it.

10 Contribution

Both team members contributed equally in this project.

References

[1] Cao, Yun, et al. "Unsupervised diverse colorization via generative adversarial networks." Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2017.

[2] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, 8â13 December 2014, Montreal, Quebec, Canada, pp. 2672â2680 (2014).

[3] Reed, S.E., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: ICML 2016, New York City, NY, USA, 19â24 June 2016, pp. 1060â1069 (2016)

[4] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. CoRR abs/1611.07004 (2016)

[5] Image retrieved from: https://www.oreilly.com/learning/generative-adversarial-networks-for-beginners

[6] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[7] Retrieved from https://www.kaggle.com/aleksandarkostadinov/image-colorization-cnn/data

[8] Retrieved from http://press.liacs.nl/mirflickr/mirdownload.html

[9] Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: ICCV 2015, Santiago, Chile, 7â13 December 2015, pp. 415â423 (2015)

[10] Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM (2001) 327â340

[11] Charpiat, G., Hofmann, M., SchÂ¨olkopf, B.: Automatic image colorization via multimodal predictions. In: Computer VisionâECCV 2008. Springer (2008) 126â139

[12] Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 649â666. Springer, Cham (2016).

[13] Deshpande, A., Lu, J., Yeh, M.C., Forsyth, D.A.: Learning diverse image colorization. CoRR abs/1612.01958 (2016)

[14] http://www.robots.ox.ac.uk/ vgg/data/flowers/17/

[15] https://github.com/TengdaHan/Image-Colorization/blob/master/asset/image-colorization-deep.pdf