Auto Colorization with cGANs

Yih CHENG
Department of Electronic Engineering
ychengaw@connect.ust.hk

Yao ZHANG
Department of Computer Science
yzhanghf@connect.ust.hk

Abstract

Image translation has been one of the most popular topics in computer vision, and approaches aiming to study the mapping function between input images and output images have been one of the main focuses toward this problem. In this project, we will target the colorization of grayscale flower images with the help of large, deep conditional GAN (cGAN) architecture. A cGAN consists of one generator and one discriminator. Working in the CIELAB color space, the generator generates the "ab" channels of an input image of "L" channel. After concatenation of "L" and "ab" channels, the colorized version of the image can therefore be retrieved. The discriminator is used to check whether the generated colorful image is real(ground-truth) or fake(generated). To find a more efficient and high-quality method, we implemented two versions of generator. The first one uses a U-net structure that includes 8 U-net blocks for the encoder and 8 U-net blocks for the decoder. Each encoder U-net block includes a convolutional layer followed by a batch normalization layer and an activation layer. Dropout is also implemented. For the decoder U-net blocks, each includes a deconvolutional layer followed by a batch normalization layer and a "ReLu" activation layer. In addition , for the last U-net block in the decoder, the "ReLu" activation layer will be replaced by a "Tanh" activation layer. In the second implementation of our generator, we utilized the Dynamic U-Net from fastai library, which is a structure of "U-Net" with a "ResNet18" backbone from torchvision library. For the discriminator, it is composed of three Conv-BatchNorm-LeakyReLU blocks. In the first block, no normalization is used and in the last block, neither normalization nor activation function is implemented. Last but not least, to utilize our project with GPU, we trained our models on Google Colab.

1. Introduction

Nowadays, a large number of precious photos full of historical meanings are in black and white colors, and for some colored cultural relics, the colors still faded away due to improper protection. Such a situation is hard for both archaeologists and artists to retrieve the original colors as it is time consuming and requires huge efforts. However, with the help of GANs, we may automatically colorize them in an acceptable time duration, helping our understanding towards those valuable exhibits and discover the stories behind these photos. However, large scale of old photos that suit our needs is difficult to find, and therefore, we decided to feed the networks with grayscale images that have similar characteristics as the old black and white pictures.

1.1. Problem Description

Given an input grayscale image $x_i \in$[0,255]$^{m \times n \times 3}$ in domain X, we want to output the corresponding colorized image $y_i \in$[0, 255]$^{m \times n3}$ in domain Y, with m being the image height, n being the image width, and 3 representing the three RGB channels. The goal for this project is to develop a model that can learn the mapping function G_{XY} : X $\rightarrow$ Y , which is capable of generating output images G_{XY}(X) that are indistinguishable from the ground-truth images Y. We successfully implemented a cGAN model to solve this problem and details about its structure will be given later.

1.2. Results Overview

In this project, the input should be a set of grayscale flower images and the output should be a set of colorized flower images. The results overview can be seen in Figure 1.

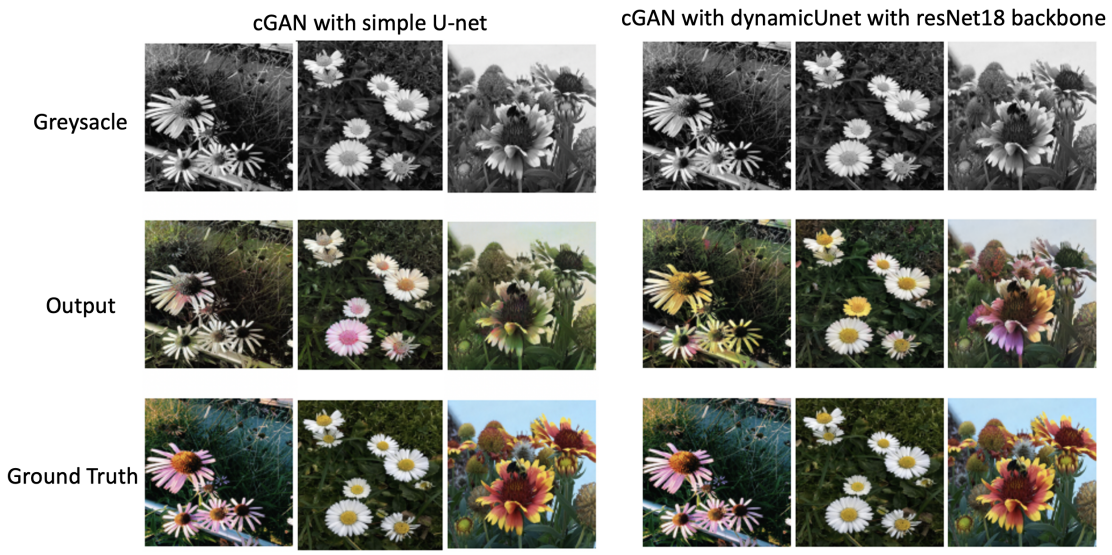

Figure 1. The output of conditional GAN

2. Related Work

Convolutional neural networks(CNN) has soared hugely in popularity ever since AlexNet won the ImageNet Large Scale Visual Recognition Challenge back in 2012. Since then, it has been one of the most common and basic ways when it comes to computer vision works. Among all computer vision problems, Image colorization has been one of the biggest topics. Several methods have been implemented to attack this problem, including Convolutional Neural Networks(CNN), which is well-suited for this task since large amount of paired data needed can be easily obtained. Approaching this task from CNN-based regression perspective have been shown to perform well without the need for human to adjust and revise large amounts of data [1, 2]. However, for regression models using L1 or L2 losses, images may end up being blur and desaturated, since CNN tries to train model to predict the average color value over all possible color values for each pixel. For images that have clear object outlines or nonconvex shapes, it may suffer and cannot produce a more determined color predictions. The introduction of Generative Adversarial Nets (GANs) provides another method towards approaching image colorization tasks. In paper *U-net: Convolutional networks for biomedical image segmentation* [3], a UNet based CNN generator along with a PatchGAN Discriminator were combined for better grayscale image colorization. The implementation of PatchGAN helps discriminate the results from the generators in a local region of images, allowing the generator to learn high frequency characteristics, while leaving the low frequency characteristics for L1 loss to capture.

Recent years, works have been done on different transformations of GAN architectures. Conditional GANs (cGANs), first introduced in *Conditional Generative Adversarial Nets* [4] submitted by Mehdi Mirza and Simon Osindero in 2014, provided an approach to solve the problem that GANs would perform poorly when trained on two different domains of datasets. It aims to tell the generator to generate images from a specific category by concatenating specific input images with random noises so that the generator is no longer deterministic. Another method approached by the paper *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* submitted by Zhu *et al.*[5] is cycleGAN, targeting tasks that do not have paired datasets. CycleGAN learns to translate an image from domain X to target domain Y without the need of paired inputs. It consists of two generators and two discriminators, with one generator converting images from domain X to Y and the other from domain Y to X. Each generator has a respective discriminator that attempts to tell apart the generated image from the ground truth. The architecture therefore forms a cycle and is named cycleGAN.

Exploring various architectures helped our team build a model to generate colorized image from grayscale images. We approached from different attempts, including a cycleGAN attempt and a cGAN attempt.

3. Dataset

The dataset we used in this project consists of a specific object category, flowers, since the color distribution will be more similar than images from random object categories. For instance, flowers will more likely be in colors of yellow, red or white, compared to colors such as black or blue, which however might be quite common in images of sneakers. To simplify the task and make the process more efficient, we first converted the images from the original *RGB* color space into *CIELAB* color space and split them into *L* channel and *ab* channel. The reason of switching to *CIELAB* color space rather than working on *RGB* color space is that if we trained in the *RGB* color space, there will be $256^3 \approx 16,800,000$ possibilities for every pixel for the generator to predict. However, in the *Lab* color space, the possibilities for every pixel will greatly decrease to $256^2 \approx 65,000$, which will make training and predicting much easier. After converting to *CIELAB* color space, we then converted *L* channel images into $x_i' \in$[-1,1]$^{m \times n \times 1}$ and *ab* channel images into $y_i' \in$[-1,1]$^{m \times n \times 2}$. In order to reproduce the original image, we will need to convert the concatenated *L* channel input and the generated *ab* channels output from *CIELAB* color space back to *RGB* color space.

3.1. Image Preparation

With the help of Google Image Download script [6], we successfully downloaded 2200 flower images and removed non-ideal and repeated images by manually going through the whole dataset. After manual revision, the dataset was then reduced to 2000 flower images. Since the original images are of *RGB* color space with different heights and widths (e.g. $325 \times 424 \times 3$), we reshaped them into $256 \times 256 \times 3$ square images with *Pillow.Image* and *torchvision* library. After that, with the help of the *skimage.color* library provided in scikit, we split the images from its original *RGB* color space into *CIELAB* color space.

3.2. Image Pre-processing

The pixel value of the image in*CIELAB* color domain are different between different channels. For *L* channel, all pixels are integer values ranging between 0 to 100, whereas for *ab* channel, they range between -127 to 128. We normalized all the pixel values to range in between -1.0 and 1.0 and then zero-centered them in order to improve the training result since activation functions will be much more sensitive to weight changes especially around zero. Apart from normalization, we also applied regularization with *RandomHorizontalFlip* provided by *torchvision.transforms*.

4. Methods

Our project mainly approached image colorization task through contional GAN. Our GAN is conditioned on grayscale images. We are going to explain the mechanism of it, including the formulation and the loss function below.

4.1. Formulation

Conditional GAN models consist of one generator and one discriminator. The generator acts as a mapping function G : X $\rightarrow$ Y. The discriminator D, adds constraints to G, which helps G generate indistinguishable results G(X). In short, we want D to tell us that G(X) = Y. Both G and D are trained together by using an adversarial loss denoted by $L_{cGAN}(G, D)$. Besides of this, we summed this loss function with L1 Loss of G(X) and Y to further help the models and introduce some supervision in our task. The adversarial loss helps train high frequency features and leaves the low frequency features for the L1 loos. Detailed visualization of the structure conditional GAN model is shown in Figure 2.

4.2. Adversarial Loss

We applied adversarial loss between the mapping function generator and the corresponding discriminator. Considering x as the grayscale image, z as the input noise, and y as the ground truth, the objective loss function is as below:

$$L_{cGAN}(G, D) = \mathbf{E}[log(D(x, y))]$$

$$+\mathbf{E}_{x,z}[log(1 - D(x - G(x, z)))]$$

Our generator G tries to generate an image $G(x, z)$ which should have little difference with the corresponding ground-truth image y, minimizing the objective loss function. While at the same time, our discriminator D tries to distinguish between the generated image $G(x, z)$ and the corresponding ground-truth real image y, maximizing the objective loss function. In other words, the objective loss is $min_G max_D L_{cGAN}(G, D)$.

4.3. L1 Loss

The adversarial loss function alone is already capable of producing colorful images, but to further improve our model and introduce some supervision in

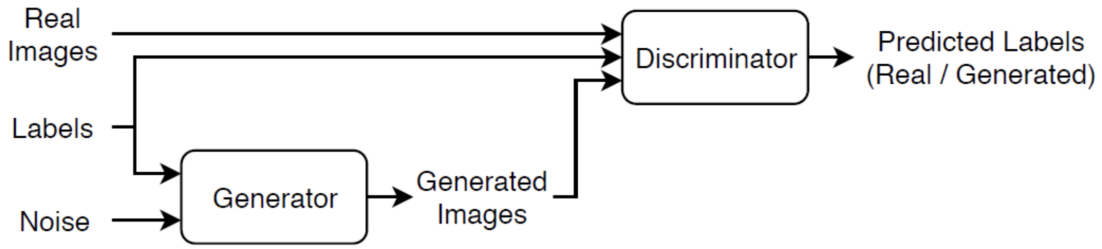

Figure 2. Structure of conditional GAN

our task, we combine this loss function with an L1 Loss of the predicted colors compared with the actual colors as well. The L1 Loss function is as followed:

$$\mathbf{L}_{L1}(G) = \mathbf{E}_{x,y,z}[||y - G_{x,z}||_1]$$

One thing worth noticing is that if we only use L1 loss, the model will still be able to learn to colorize the images but the colors will be more conservative. For example, if the ground truth is a green flower, L1 loss will calculate the difference between $G(x, z)$ and ground truth. Suppose the next flower image has ground truth color of blue, the L1 loss will then try to minimize between the $G(x, z)$ and blue flower. From the feedback of L1 loss, when the model faces a similar grayscale flower image the next time, it will know to output a color that is simultaneously close to blue and green, which will end up being color cyan. However, a flower of color cyan doesn't exist. If we train the L1 loss through a large dataset, we might end up producing conservative colors since L1 loss tries to minimize the output with every learned color, which we do not want.

4.4. Full Loss

Combining the L1 Loss and the adversarial loss, we now obtain the Full loss function for the model:

$$L = \min_G \max_D L_{cGAN}(G, D) + \lambda \mathbf{L}_{L1}(G)$$

4.5. Our Model

Our model consists of a generator and a discriminator. We used a "patch" discriminator as our discriminator model. In order to compare between different generator structures, we implemented two structures of our generator model, with the first one being U-Net and the second one being still U-Net but with a ResNet backbone. Detailed implementation is provided below.

4.5.1. Patch Discriminator

In our model, a "Patch" Discriminator is implemented. For this patch discriminator D_Y, it contains 5 blocks, with each block containing a convolutional layer, followed by a batch normalization layer if it is an inside block, and a *Leaky ReLu* activation layer if it is not the first block. The difference between using the patch discriminator versus a standard discriminator is that a standard discriminator takes in a 256×256 input image and outputs a scaler, whereas a patch discriminator maps the image from 256×256 to 70×70 (in our case) array of outputs X. Each output $X_{i,j}$ signifies whether patch *i, j* of the image is ground truth or generated image. To make things simple, our patch discriminator chops the input image into 70×70 patches, makes a big batch of these patches, and sends each patch through the discriminator. Implementing a patch discriminator can perform better for our model since every local spaces matter and we do not want to judge on a simple scaler whether the image is ground truth or generated.

4.5.2. U-Net Generator

U-Net is an encoder-decoder style architecture with long skip connections between mirrored layers in encoder and decoder blocks. Our encoder down-samples the input image to a smaller feature map through convolutional layers, and the decoder up-samples the feature map using deconvolutional layers. And for every result of a deconvolutional layer, the corresponding result of the mirroring convolutional layer is added to it. The lower encoder blocks or convolutional layers contains lower level features since the feature map is chopped down throughout the whole encoder process, and adding "long jumps" to the mirrored result of every deconvolutional layer helps the model maintain higher levels of features extracted

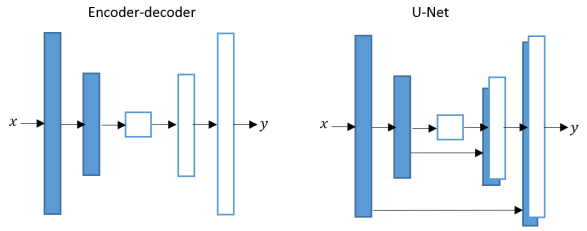

Figure 3. Structures of Encoder-Decoder and U-Net.

previously from higher encoder blocks or convolutional layers. The idea of encoder-decoder and "U-Net" is shown in Figure 3. Our generator model consists of 8 encoder blocks and 8 decoder blocks. For encoder blocks that are not the most inner block or most outter block, the structure is simply a convolutional layer followed by a Batch Normalization layer with a *Leaky ReLu* activation function. For the most outter encoder block, it contains only one convolutional layer with *Leaky ReLu* activation function, and for the most inner block, it contains only one convolutional layer with a *ReLu* activation function. For all decoder blocks except most outter block, the structure is a deconvolutional layer followed by a Batch Normalization layer with a *ReLu* activation funcion. Worth noticing is that the 3 decoder blocks after the most inner decoder block also include a dropout layer with a 0.2 dropout rate to prevent overfitting. For the most outer decoder block, it contains only one deconvolutional layer with *tanh* activation function. Skip connections between mirroring layers are added together as mentioned.

4.5.3. U-Net Generator with ResNet Backbone

For the second version of Generator, we further try to combine the "Long Skipping" and "Short Skipping" together to see if the model can achieve a better result. Before ResNet, it was a common belief that the deeper the layer, the better the training results. However, this was proofed to be wrong in *Deep Residual Learning for Image Recognition* [7]. The team proposed a new way of mapping, which is by stacking the nonlinear layers *X* to the original mapping function *F(X)*, forming the identity mapping function *H(X) = F(X) + X*, which can also be indicated as short skipping. The identity mapping function is more sensitive to output changes, making it easier to train according to gradient descent. Eventually, this makes the training much easier since the contents that the model needs learn is less and they are easier to learn. In our model, we apply this concept into each block defined in the above U-Net Generator.

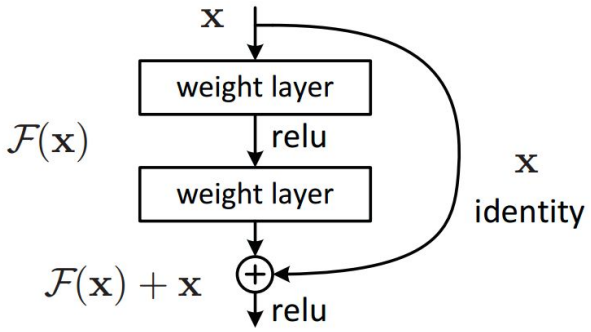

Figure 4. Structure of simplified resNet block

Using the ResNet18 model in torchvision library and DynamicUNet from fastai library, we hypothesize that this model can provide even better and faster results compared to that of a standard CBR U-Net. And the structure of a simplified resNet block is shown in Figure 4.

5. Experiments

5.1. Discussion on Inner Structure

After implementing the first version of Conditional GAN, we successfully generated some reasonable colorized images of flowers as shown in Figure **??**. Although this U-Net model maintains some basic understanding of some objects in images such as different shapes of petals should have different colors and grass should be green, its output is far from something appealing, for instance it cannot decide on the color of rare flowers. Also, this model generates some color spillovers and circle-shaped region colors, which means the boundaries are not precise. Later on, inspired by the Residual network we learned during the class, we came across the idea of combining U-Net and ResNet together for our model to achieve a more precise result. We hypothesize that adding short jumps can furthermore preserve some detail features. After all, this structure can hold both the advantages of "long jump" brought by U-Net and "short jump" brought by ResNet. Referring to *UNet with ResBlock for Semantic Segmentation* [8] and *Image-to-Image Translation with Conditional Adversarial Networks* [9], we are able to construct a similar but more basic model, and the results are shown in Figure **??**. We can see that the quality of the output images improve even more and the boundaries are sharper and more precise. On the other hand, it took only xx times of the training time of the standard CBR U-Net generator, which is also what we expected.

5.2. Failed Models

During our experiment, we also tried to implement the CycleGAN architecture with the help from *SINGLE IMAGE COLORIZATION VIA MODIFIED CYCLEGAN* [10], which eventually didn't meet our expectations. For CycleGAN architecture, it makes use of an additional inverse mapping function G_{YX} : Y $\rightarrow$ X that furthur maps the generated image back to the original domain *X*. Also, in additional to adversarial loss such as that in conditional GANs, it also adds cycle consistency loss to minimize the difference between $G_{YX}(G_{XY}$ (X)) and ground truth *X*. We tried training it with 100 epoches, but the results are just marginally acceptable and far from satisfying compared to that of cGANs and they are shown in Figure 5. However, it took us double time for training this result compared to standard U-NET generator in cGAN architecture. We believe that given optimized conditions, it can eventually produce results that is similar to that of cGAN, but the training time can increase to over hundreds of hours, which is not achievable as we are training through Google Colab and the GPUs they provide have time limitations.

It would be unfair to claim that cycleGAN cannot perform well in our image colorization tasks at this momet, as we have limited time to tune the hyperparameters and limited resources. In fact, we hypothesize that cycleGAN models can produce even better performances considering its architecture, but they are very difficult to train, as compared to cGANs. Some related research of this area showing cycleGAN is capable of this task is *SINGLE IMAGE COLORIZATION VIA MODIFIED CYCLEGAN* [10] and *Colorizing Near Infrared Images through a Cyclic Adversarial Approach of Unpaired Samples*. [11]

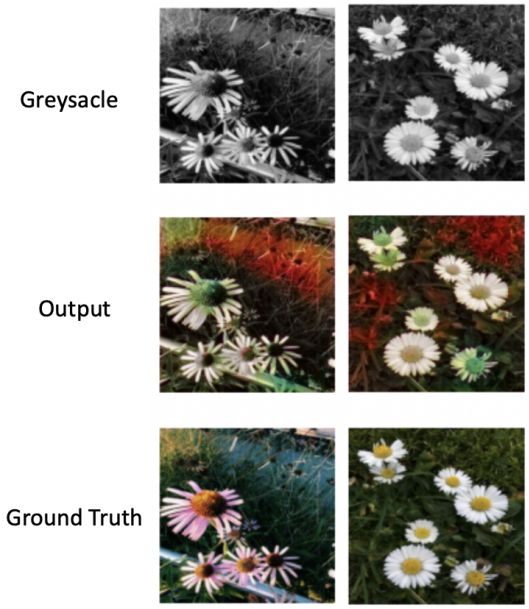

Figure 5. The output of conditional GAN.

5.3. Coloring Flower Sketches

To test our model in real case scenarios, we fed some flower sketches (which are one kind of grayscale images) drawn by ourselves into our Conditional GAN model. It turns out that the generated colored images look quite real, suggesting that the trained model has successfully achieved our goals and objectives we proposed at the beginning. Some sample results are shown in Figure 6.

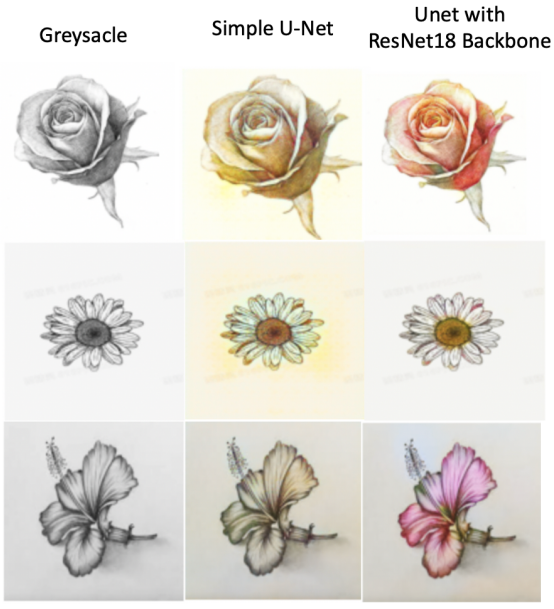

Figure 6. Test with Sketch of Flowers

6. Conclusion

In this project, we have tried to apply different kinds of Generative Adversarial Networks to solve the problem of auto colorization of grayscale images. As the results suggested, within the same period of

time, conditional GAN models show better performance than classical GANs models and cycleGAN models. By using cGANs with proper inner structure, realistic results can be obtained with high efficiency and quality. By participating in this project, we learned more about advantages and disadvantages of different types of GANs, which can greatly helped us choose between different architectures for specific tasks in further learning. In addition to that, we also gained practice and valuable experiences in training and babysitting neural networks, which is absolutely beneficial for ourselves and our future. We now have a deeper understanding about Generative Adversarial Networks models and strengthened the ability to solve realistic problems using studied knowledge.

6.1. Further Improvement

We have concluded that conditional GAN is easier to retrieve a realistic image within a limited time duration, whereas the cycleGAN is much more difficult to train but might provide a better diversity and more subjective results. We expect that with better inner structure design and sufficient time, cycleGAN can be more capable than what we concluded from this experiment. In addition, we are able to implement unpaired training as of cycleGAN, which may be better in reality, as most of the grayscale photos do not have its original colorized version, which is a must when it comes to conditional GANs. Apart from this, our ultimate goal is to develop a model that can automatically fill colors for black and white photographs taken in the old days, which contains much more variety of objects rather than just flowers and scenery. Therefore, for further improvement, it is a must for us to get some more datasets from a variety of object categories to train, evaluate, and improve our model.

References

[1] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *Proceedings of the IEEE International Conference on Computer Vision*, p. 415–423, 2015.

[2] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," in *ACM Transactions on Graphics (TOG)*, p. 35, 2015.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," pp. 234–241, 2015.

[4] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[5] J.-Y. Zhu, T. Park, P. Isola, and A. A.Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[6] sczhengyabin, "Image-downloader."

[7] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep residual learning for image recognition," 2015.

[8] N. Singla, *UNet with ResBlock for Semantic Segmentation*, 2019.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.

[10] X. Yuxuan, J. Aiwen, L. Changhong, and W. Mingwen, "Single image colorization via modified cyclegan," 2014.

[11] M. Armin and S. Angel D., "Colorizing near infrared images through a cyclic adversarial approach of unpaired samples," in *Computer Vision and Pattern Recognition (CVPR), 2019 IEEE Conference on*, 2019.