Colorization Korean War pictures Using GAN

Muhammad Ridhwan Bin Mahadi
20191203

Suh Ki-soo
20201087

Kim Ji-sun

Abstract

Colorization is a where we take a black & white or grayscale images and fill it up with colors. Colorization are commonly use to bring old black & white images to life as we can see how the past look like. So, we decided to focus on Korean war colorization. We used GAN which can produce good results quantitatively and qualitatively. We used World War 2 colorized images as our inputs because they are closely related and it can lead to near original color quality.

1. Introduction

AI is very closely connected to our modern lives. Among them, artificial neural networks have been spotlighted as AI configurations with outstanding problem-solving skills since Team Super Vision won an overwhelming difference with deep neural networks (DNN) at ILSVRC-2012. [1] According to an AI Today AI Tomorrow survey of 3,938 people conducted by Northstar Research at the request of ARM, 61% expected AI or automation to lead society in a better direction, and only 22% felt negative about it. [2] Most people think that when they hear the word AI or artificial neural network, they are positive and can achieve something that humans cannot do. And one of the results of that is Colorization.

Automatic colorization is an area of research that has lots of potentials in applications. Specifically, we will take a look at Korean War picture colorization. The period during which the Korean War took place was Jun 25, 1950 – Jul 27, 1953. And we can say that there is no color picture at that time. If so, how should we proceed with the coloration and which sample pictures should we use to proceed with the coloration? This is the crux of this research.

We envision using Generative Adversarial Network (GAN). [3] This is because we thought that coloring Korean War pictures was more of a creative process of creating new pictures than restoring them. And we thought that GAN was the most appropriate for this process. The results of this study show that colorization GAN trained by only the images associated with the topic can show accurate result, which can help determine whether it is efficient to build a GAN for a particular purpose.

2. Related Work: DeOldify

DeOldify is a Colorizing and Restoring Photos & Video program that has been available since 2018. It boasts nearly the best performance among the Colorizing programs currently being deployed. The program was created using NoGAN, a type of GAN, and a pre-trained model can be obtained from Jantic's github, https://github.com/jantic/DeOldify. Our work will be evaluated by comparing with this DeOldify. Although this DeOldify is the best program, it may not be suitable for a special situation called Korean War because it is designed to color all types of images.

3. Methods

3.1. ResNet34

ResNet (Resident Networks) is a classic neural network used as a backbone for many computer vision tasks. [4] The model won the ImageNet Challenge in 2015. ResNet34 has weights and biases that have been learned from the dataset and represent the capabilities of the learned dataset. The learned functions can often be transferred to other data. The pre-trained model is beneficial to us for many reasons. We can save time using pre-trained models.

In this work, ResNet34 will be used as the basis for U-net. The generator is built on U-net, where the classification of basic objects will already use what

ResNet34 has been trained on.

3.2. U-Net

U-Net is a convolution neural network developed for image segmentation at Freiburg University in Germany. [5] This network is based on the fully convolution network of Long, Shelhamer and Darrell, working with fewer training images and the architecture modified and extended for more accurate results.

The main transformation point is to complement a typical contract network with a continuous layer where pooling operations are replaced by upsampling operators. Therefore, these layers increase the resolution of the output. U-Net have a large number of functional channels in the upsampling section. This allows the network to propagate context information to higher resolution layers. As a result, the expansion path is somewhat symmetrical to the contraction and creates a U-shaped architecture. The network uses only the valid parts of each convolution without a fully connected hierarchy.

4. Dataset

4.1. Type of Dataset: World War II

Most AIs are based on data. And these data should be deeply correlated with AI's goals. David Silver said, "We trained the policy network p_σ to classify positions according to expert moves played in the KGS data set. This data set contains 29.4 million positions from 160,000 games played by KGS 6 to 9 dan human players." [6] AlphaGo, the most famous artificial neural network and the hottest in Korea against Lee Se-dol on March 9 and 10, 2016, was trained using appropriate data in both the early and mid-term periods of the production process. As such, it is very important to use the appropriate data in learning. In this study, especially since there were few color photos of Korean War, color photographs were needed to replace them. When examining the equipment and weapons used in the Korean War, they were very similar to those used in World War II. [7][8] In addition, unlike the Korean War, which has few color pictures, they have a large number of colored photos. So, we decided to use the pictures of World War II as dataset.

4.2. Collecting Dataset

The Icrawler package was used to collect dataset. We used Torch version 1.4 and Torchvision 0.5.0, due to errors in the latest versions.

```
!pip install "torch==1.4" "torchvision==0.5.0"
```

As mentioned earlier, the data was downloaded using the Icrawler, but the Google search html code changed while the Icrawler was not updated recently. So we changed the data parsing part slightly. The modification was based on Harry Wong's code on the Icrawler github. [9]

Since then, data extractions have been made using color="color" from the filter to take color images of World War II. In addition, because Google has been able to load up to 700 images at a time, we split the period every year and downloaded the pictures. Since then, we deleted irrelevant photos.

```
class GoogleImageCrawler(Crawler):

from icrawler.builtin import GoogleImageCrawler

google_crawler = GoogleImageCrawler(feeder_threads=1, parser_threads=1, downloader_threads=4,

storage={'root_dir': path})

filters = dict(color='color')

google_crawler.crawl(keyword=’World war 2', filters={'date': ((2020, 1, 1), (2020, 12, 1))}, max_num=1000, file_idx_offset=0)

google_crawler.crawl(keyword= World war 2', filters={'date': ((2019, 1, 1), (2019, 12, 31))}, max_num=1000, file_idx_offset='auto')

…
```

4.3. Convert to black and white

Black-and-white photographs were made using decolorizer after defining decolorizer using PIL packages. And we labeled pictures with the original color pictures.

```
from PIL import Image, ImageDraw, ImageFont

class decolorizer(object):

    def __init__(self, path_lr, path_hr):

        self.path_lr = path_lr
```

```
        self.path_hr = path_hr

    def __call__(self, fn, i):

        dest = self.path_lr/fn.relative_to(self.path_hr)

        dest.parent.mkdir(parents=True,
exist_ok=True)

        img = PIL.Image.open(fn).convert("L")

        img.save(dest)
```

5. Experiment

5.1. Setup

First, we need to decolorize the images so create a function to turn the images to black & white. Because the process to decolorize of images can take a while, we will pass a 'parallel' function where it can run several processes in parallel. This will save a lot of time.

5.1.2 Pre-train generator

Second, we pre-train the generator on the training set which we have converted from colored to grayscale. We set the batch size 32 and size 128 pixels. We use the get_data() to create the data bunch where it will be use to create the leaner. To restore the image from learning from the original image, we going to use UNet. We going to use this UNet to build the need for GANs. As per the requirement, transformed and normalized is used. In the normalize method, we are using the ImageNet stats because we will use a pre-trained model. We set the weight decay(wd) = 1e-3 and y_range (sigmoid function) = (-3.,3.). To compare the output with the original image we defined loss function as MSELossFlat(). After the UNet learner process is done, we will now fit the model. Since the fitting model happens with freezing, to use transfer learning, we will unfreeze the pre-trained part of UNet.

Figure 1 – generated images

5.1.3 Pre-train discriminator

Third, we will use the generator generated images and the original images to train the discriminator. We created a path and folder to save our generated images. Next, we define the get_crit_data classifier and define learner for the classifier. While defining the learner, gan_critic is use to give a binary classifier suitable for GAN. accuracy_thresh_expand is used because there are slightly different in architecture and loss function. We call create_critic_learner function to create our discriminator.

epoch	train_loss	valid_loss	accuracy_thresh_expand	time
0	0.706609	0.688924	0.560000	00:14
1	0.699407	0.688112	0.586897	00:10
2	0.690482	0.976069	0.614483	00:11
3	0.693045	0.689731	0.547586	00:10
4	0.686882	0.584899	0.595172	00:10
5	0.643912	0.505988	0.715862	00:10
6	0.579641	0.394433	0.788966	00:10
7	0.509817	0.170753	0.960690	00:10
8	0.424571	0.090227	0.962759	00:10
9	0.356108	0.093840	0.964138	00:10

Figure 2 - learn_critic.fit_one_cycle(10, 1e-3)

After 10 epochs, the discriminator was able to tell if the image is generated with 96% accuracy. We have a good learner in differentiating generated images from original images.

5.1.3 GAN

Now we train pre-trained generator and pre-trained discriminator. We pass GANLearner in our generator and discriminator. We set our weights_get = (1.,100.). The value 100 gives the generator huge incentive to learn generating images close to ground truth. And we set betas = (0.,0.5) because decreasing betas is effective in training GANs to prevent the discriminator learn too fast and the discriminator loss drop to zero soon leading to the failure of learning for the generator. [10] After training, here are the results.

Figure 3 – GAN generated images

5.2 Comparing with DeOldify

We compare one of our GAN images with the DeOldify to see how good are our training.

Figure 4 – Black and white image

Figure 5 – GAN

Figure 6 - Deoldify

After comparing, we can see that our GAN generated image is still not as good as Deoldify. Deoldify generated images have good colors and contrast. Our color tones are lighter than Deoldify.

6. Conclusion

In this project we colorize black & white images in our case War images using GAN. As our GAN generated images is not as good as the Deoldify due to our small-scale dataset, we think that we can increase our image dataset size so can our generator/discriminator/GAN can train better to give a prediction and result. Another room for improvement is finding the perfect learning rate, weight and increasing the number of epochs.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, 2012 p. 1-9.

[2] ″AI Today AI Tomorrow”(2020.11.26. 15:00 accessed), https://www.arm.com/resources/report/ai-today-ai-tomorrow.

[3] Goodfellow, Ian et al., “Generative Adversarial Networks.”, NIPS 2014, 2014, p. 2672–2680.

[4] ″Resnet34”(2020.11.27. 2:00 accessed), https://www.kaggle.com/pytorch/resnet34

[5] Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas, "U-Net: Convolutional Networks for Biomedical Image Segmentation.", arXiv:1505.04597, 2015.

[6] Silver, David et al., "Mastering the game of Go with deep neural networks and tree search". Nature. 529, 2016, p. 484-489

[7] Suciu, Peter, Stuart Bates, “Military Sun Helmets of the World.”, Uckfield, UK: Naval and Military Press, 2009.

[8] Huston, James Alvin, “Guns and Butter, Powder and Rice : U.S. Army Logistics in the Korean War.”, Selinsgrove, PA: Susquehanna University Press, 1989.

[9] Harry Wong, “Google Crawler is down #65”(2020.4.16.)(2020.12.15. 5:00 accessed), https://github.com/hellock/icrawler/issues/65

[10] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. CoRR, abs/1606.03498, 2016