Automatic Colorization

Minghui Zhou (mzhou35@ucsc.edu)
Pengyang Zhou (pzhou15@ucsc.edu)

April 11, 2019

Abstract

The goal of this project is to investigate into the automatic image colorization problem which converts a gray-scale image to a colorful one. We've built a Convolutional Neural Network (CNN) and a Generative Adversarial Networks (GAN) based on the recently state-of-art colorization algorithms. In our project, different elements of each paper's method are combined to do the colorization. We also conducted the experiments with various architectures, loss functions with the goal of producing plausible, natural, vibrant colorizations on grayscale images.

1 Introduction

When your grandma shows you a black-and-white picture of her teenage, it's hard to think a young beautiful young lady in a red dress with joyfulness. A black-and-white photograph will easily make you feel distant; however, colorization could bring in more life and characters to these moments. Colorization of old photos or movies has been recognized as highly laborious and tedious, but the results are often amazing[1].

Previous colorization methods can be roughly divided into two categories: scribble-based colorization and example-based colorization [1]. The scribble-based methods typically require substantial efforts from the user to provide considerable scribbles on the target gray-scale images, which is very time-consuming. And the example-based methods which transfer the color information from a similar reference image to the target gray-scale image, reduce the burden from user. However, sometimes, it's hard to find a similar image which is suitable for the target. Also, the performance of output highly depends on the selected reference image(s).

Therefore, we investigated the image colorization problems and conducted a survey to explore the state-of-art deep learning methods in this field. After that, we also implemented two different methods based on CNN and GAN, and ran experiments using the same dataset which is from Places365[2].

[1] https://www.reddit.com/r/Colorization/
[2] http://places2.csail.mit.edu/

2 Methods

2.1 CNN Based Colorization

Larsson et al. [5], Iizuka et al. [3] and Zhang et al.[10] used different CNN architectures and loss functions and trained models on ImageNet and Places365.

	Iizuka et al.	**Larsson et al.**	**Zhang et al.**
Model Architecture	Two-stream convolutional network	Hypercolumn approach	Single-stream convolutional network
Training Dataset	Places365	ImageNet	ImageNet
Loss Function	Regression (mean squared error)	Classification (cross-entropy)	Classification (cross-entropy)

Figure 1: comparison of these CNN algorithms from these three paper [10]

In their algorithms, they both use the LAB color-space, since its L channel includes lightness information which can be used as input grayscale image, and the ab channel has the image color information which is our predicting result. After concatenated L and ab channel together, we can get the colorized image.

2.1.1 CNN proposed by Zhang et al

In this paper, the authors notice that the biggest challenge of colorization is that the color prediction inherently multimodal – many objects can take on several plausible colorizations[10]. Therefore, the loss functions which assume uni-model outcome are not suitable, such as Square Error Loss. These losses are inherited from standard regression problems, where the goal is to minimize Euclidean error between an estimate and the ground truth. To appropriately model the multimodal nature of the problem, they predict a distribution of possible colors for each pixel. Furthermore, they re-weight the loss at training time to emphasize rare colors. Finally, they produce a final colorization by taking the annealed-mean of the distribution.

In summary, they transfer the colorization problem to the multinomial classification problem:

- Divide the output ab space into discrete bins of size 10, as in Figure 2.
- Predict a distribution of possible colors for each pixel by using multinomial cross entropy loss.

$$L^{\widehat{Z},Z} = -\frac{1}{HW}\sum_{h,w}\sum_{q} Z_{h,wq}\log(\hat{Z}_{h,w,q}) \tag{1}$$

- Produce a final colorization by taking the annealed-mean of the distribution.

$$\mathcal{H}(Z_{h,w}) = \mathbb{E}[f_T(Z_{h,w})], \quad f_T(Z) = \frac{\exp\left(\log(z)/T\right)}{\sum_q \exp\left(\log(z_q)\right)} \tag{2}$$

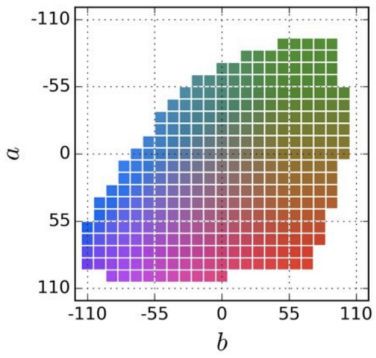

Figure 2: Quantized ab color space

The CNN network architecture they built is as Figure3. Each conv layer refers to a block of 2 or 3 repeated conv and ReLU layers, and there is no pooling layer, image size is changed through spatial downsampling or upsampling.

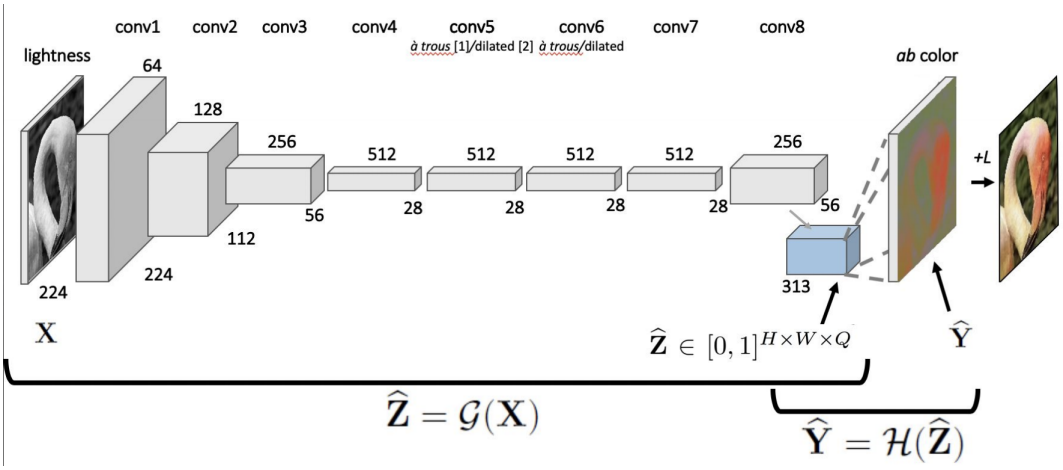

Figure 3: The network architecture [10]

The contributions in this paper are that they design a appropriate objective function that handles the multimodal uncertainty of the colorization problem, and that they set a high-water mark on the task by training on a million color photos over several weeks.

2.1.2 CNN proposed by Lizuka et al

Lizuka et al. propose an end-to-end network that jointly learns global and local features for automatic image colorization. Their network includes two branches, one extracts local features and the other extracts global features from the image, and the network is composed of four sub-networks, Low-Level Feature Network, Mid-Level Features Network, Global Features Network and Classification Network as shown in Figure 4.

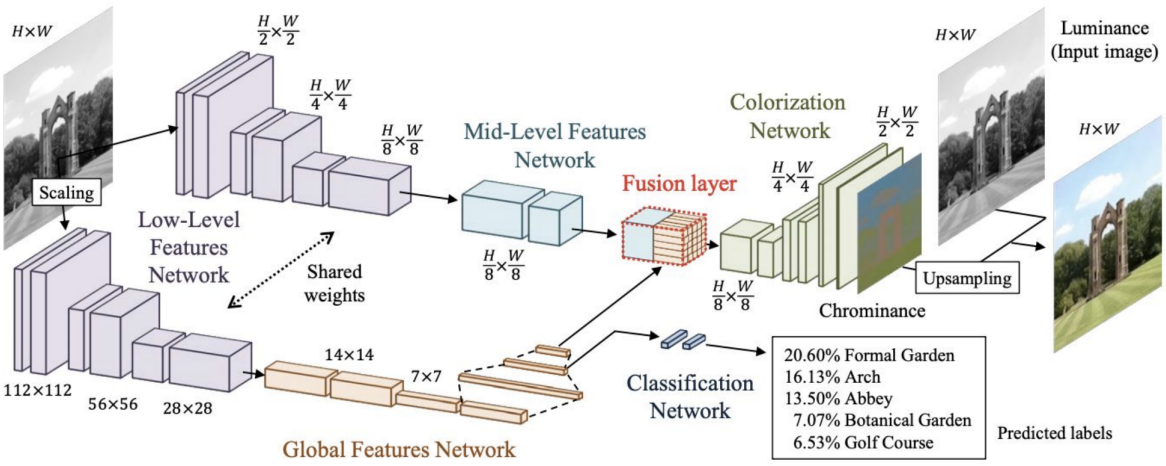

Figure 4: The network architecture [3]

- Low-Level Feature Network
  It extracts shared low-level features such as edges and corners. Instead of using max-pooling layers, it uses convolution layers with increased strides to reduce size. (If padding added, the output is effectively half the size of the input layer with $stride = 2$)

- Global Features Network
  It further processes low-level features with 4 convolutional layers followed by 3 fully-connected layers. It computes a global 256-dimensional vector representation of the image.

- Mid-Level Features Network
  It further processes low-level features with 2 convolutional layers. Low-Level and Mid-level features network are combined together as a fully CNN to get details of local features in the image.

- Fusion Layer
  Fusion layer incorporates the global features into local features. It effectively combines the two kinds of features and obtains a new feature map.

  – **Global features:** 256-dimensional vector.
  – **Local features:** $H/8 * W/8 * 256$ matrix.

$$y_{u,v}^{fusion} = \sigma\left(\mathbf{b} + W\begin{bmatrix} y^{global} \\ y_{u,v}^{mid} \end{bmatrix}\right) \tag{3}$$

- Classification Network

  – Convolution layers + Upsampling layers (nearest neighbor technique).

– Output layers: convolutional layer with a sigmoid transfer function.
– Output: Chrominance (a, b channel) + Luminance (Lightness channer) = colorful image

In this paper, the authors also proved that the Mean Squared Error couldn't learn global content properly. They improved the optimization by also training for classification jointly with the colorization. After added 2 FC layers to predict the class label of image, they combine MSE loss for colorization, cross-entropy loss for classification together as below:

$$L(y^{color}, y^{class}) = \left\| y^{color} - y^{color,*} \right\|_{FRO}^{2} - \alpha \left(y_{l^{class}}^{class} - \log \left(\sum_{i=0}^{N} \exp \left(y_i^{class} \right) \right) \right) \quad (4)$$

2.1.3 Our CNN Model

In our implementation, after tested different CNN architectures and loss functions, we built a fully convolutional (single-stream forwarding) network, inspired by Zhang et al. First of all, we pretrained a ResNet-18 classifier with grayscale image which not only speeds up the training process but also extracts quantity of information in the lightness channel of an image.

Our final CNN network architecture is shown in Figure 5 and table 1. It consists of a ResNet classifier followed by 6 convolution layers and 1 upsampling layer.

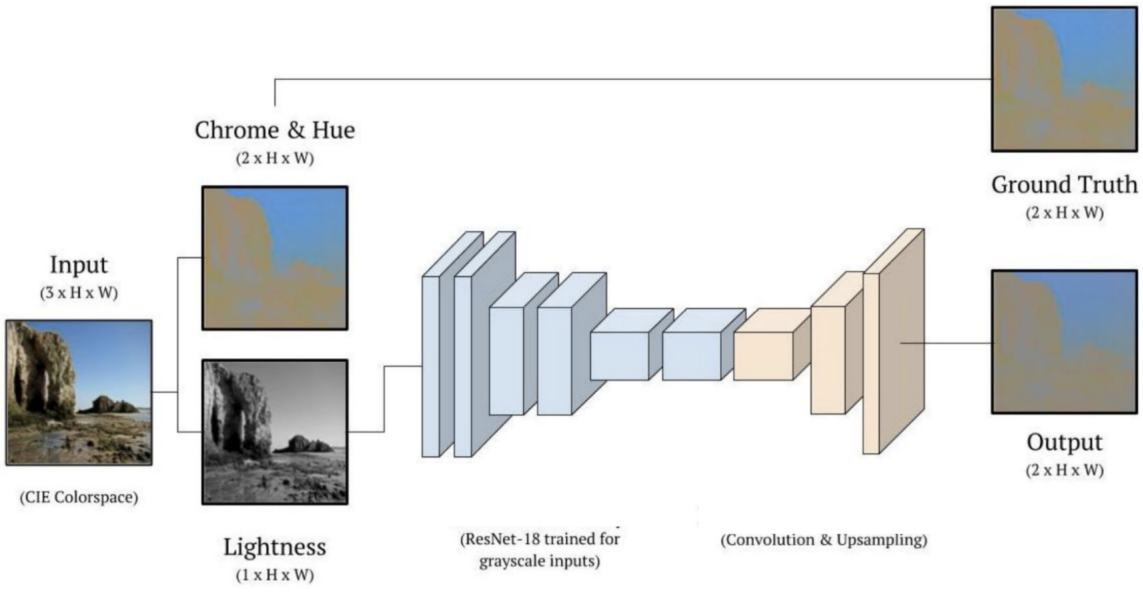

Figure 5: The CNN network architecture we used

We tried mean absolute error (L1), mean squared error (L2) and cross-entropy loss functions, finding that cross-entropy loss functions produced more vibrant colorizations. we shows the results in the following section.

Type	Kernel	Stride	Output
conv.	4*4	2	128
conv.	3*3	1	128
conv.	3*3	1	64
conv.	3*3	1	64
conv.	3*3	1	32
conv.	3*3	1	2
upSample	-	-	2

Table 1: The details information of our CNN model

2.1.4 Results and Discussion

Figure 6 show on set of the results which contains the original picture, the grayscale picture, and the colorized picture. As you may have noticed, the colorized results are not exactly the same as the original pictures. And our goal is not to recover the graycale pictures to original ones, instead, as long as it makes sense, we want them to look plausible, vibrant and natural. So the colorization results in Figure 6 are considered as acceptable.

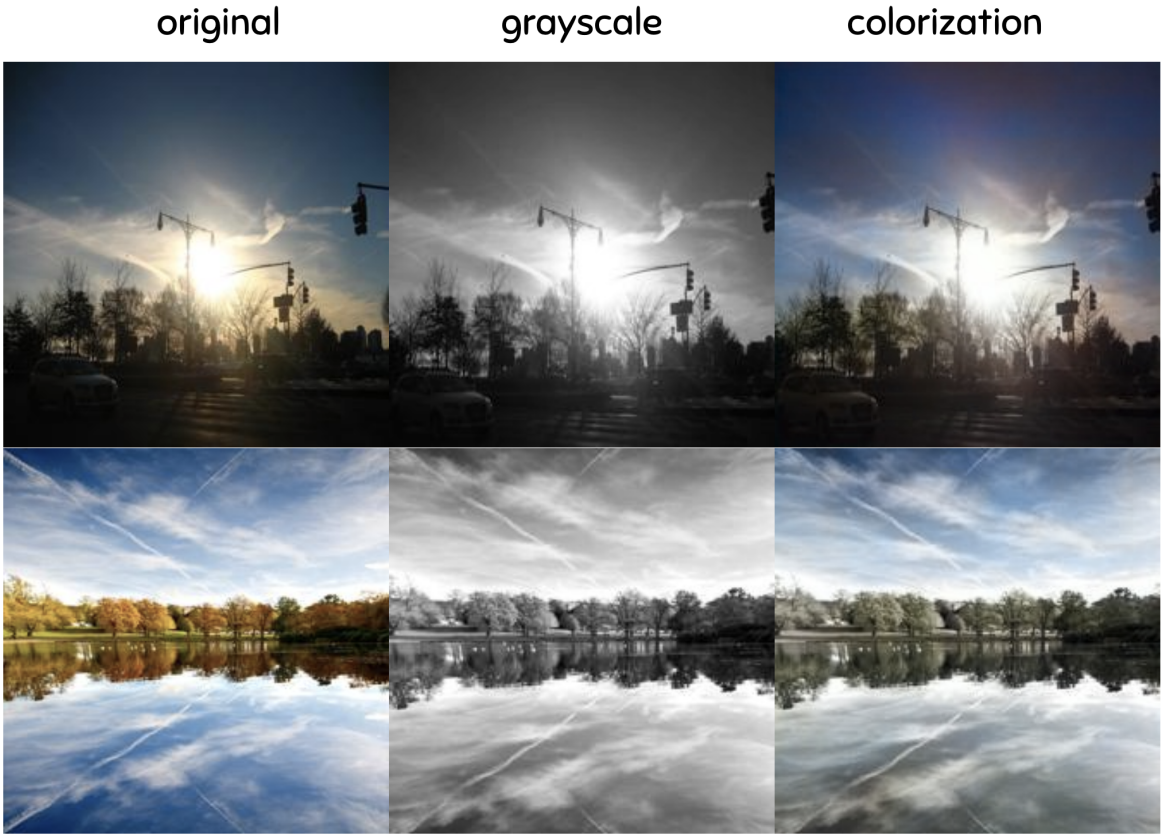

Figure 6: Colorization Result

Figure 7 shows some results which are considered by us as excellent results. For these results, if we don't put the original images side by side, people can hardly tell they're colored by machines.

Figure 7: Colorization Examples of Images in Place365

We also took some pictures on our campus near the classroom. Figure 8 show some results. The left-hand side shows as set of pictures which have grass, trees, and sky. The original picture (left) is side by side with the colored picture (right). As you can see, the results are still acceptable and make sense. The right-hand side shows some negative results generated by the machine. For example, the squirrel is colored as green, and the banana slug is colored as gray. One reason for this is the short training time of our model. The total size of the Place365 dataset is very large and usually it takes a few weeks to get some fair results. In order to speed up the training results and verify the effectiveness of our models, we chose a subset of images in Place365 for the purpose of training. This shortened the training time greatly. Animals don't appear in the dataset very often, thus, the colorizations on them are not very effective.

2.2 Colorization with GANs

Generative Adversarial Networks (GANs) is an state-of-art generative model which is proposed by Goodfellow et al.[2] in 2014. The main idea of GANs is a two-player game where the sum of the interests of both sides of the game is a constant. The two players: one person's name is the generator (G), and the other person's name is the discriminator (D). They each have their own functions. Generator is to wrap noise/sample into a realistic sample, the output and discriminator is a two-classifier (like the 0-1 classifier) which is to determine whether the input sample is true or false. The optimization

Figure 8: Example Pictures of UCSC Campus

problem for training the generator and discriminator can be expressed as: G is trained to minimize the probability that the discriminator makes a correct prediction in generated data, while D is trained to maximize the probability of assigning the correct label[7]. Mathematically, cost functions are often presented as a single minimax game problem with the value function V (G, D):

$$\min_G \max_D V(G, D) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))] \tag{5}$$

Prior and concurrent works have GANs techniques on image colorization problems. In [4], Isola et al.(2017) proposed a coloring method by comparing the color differences generated by convolutional neural networks and GAN. The model not only learns the mapping from input to output, but also learns a loss function to train this mapping. Their model fit with ill-posed problems, such as synthesizing photos from label maps, reconstructing objects from edge maps, and tinting images. In [6], Mirza and Osindero(2014) attempt to fully generalize the colorization procedure using a conditional Deep Convolutional Generative Adversarial Network (DCGAN). This approach is trained over publicly datasets such as CIFAR-10 and Places365.

2.2.1 A conditional DCGAN Model by Nazeri and Ebrahimi

In this section, we will explore the GANs proposed by [2] for colorization. The proposed GANs is trained on CIFAR-10 and Places365 [11] and we will compare the result with convolutional neural networks (CNN).

In a original GAN, the generator's input is randomly generated noise data z. However, the inputs of generator for automatic colorization problem is grayscale images instead of random noise. Therefore, we use a variant of GAN called conditional generative adversarial networks[6] to address this problem. In this approach, the generator's input is using zero noise with the grayscale input as a prior, or mathematically speaking, $G(0_z|x)$. In addition, we also modify the input of the discriminator to fit the conditional GANs network. In conclusion, the final cost functions of the proposed model are as follows:

$$\min_{\theta_G} J^{(G)}\left(\theta_D, \theta_G\right) = \min_{\theta_G} -\mathbb{E}_z\left[\log\left(D\left(G\left(\mathbf{0}_z|x\right)\right)\right)\right] + \lambda \left\|G\left(\mathbf{0}_z|x\right) - y\right\|_1 \tag{6}$$

$$\max_{\theta_D} J^{(D)}\left(\theta_D, \theta_G\right) = \max_{\theta_D}\left(\mathbb{E}_y[\log(D(y|x))] + \mathbb{E}_z\left[\log\left(1 - D\left(G\left(\mathbf{0}_z|x\right)|x\right)\right)\right]\right) \tag{7}$$

Next, we will discuss the generator and the discriminator models. First, both generator and discriminator architectures are implemented by Deep Convolutional GANs (DCGAN) [8] guidelines and this structure was also modified as a conditional GAN. Then, we also follow guideline in [4] and provide noise only in the form of dropout [9]. All applied on several layers of our generator. For generator G, the architecture is the same as the baseline. And for discriminator D, similar architecture is used as the baselines contractive path: a series of 4×4 convolutional layers with stride 2 with the number of channels being doubled after each downsampling. For convolution layers, all of them followed by batch normalization and use leaky ReLU activation with slope 0.2. For last layer, we use a convolution to map to a 1 dimensional output and use sigmoid function in order to return a probability to show real or fake. Both true colored image and fake image generated by generator concatenated with the grayscale image are input image of discriminator. The network architecture of this conditional DCGAN is shown as Fig.9 .

2.2.2 Experiments with DCGAN Model and Improvements

In the implementation and experiments with image colorization, we first followed the conditional Deep Convolutional GAN architecture, and analyzed the model by training for different epochs. Then, we improved the loss function for generator and adjusted our trainning strategies to have a better performance. At last, we compared the testing results for our trained models at several checkpoints.

Environment Setup: We use our Google Cloud Deep Learning Virtual Machine Instances for the model training and testing, with a NVIDIA Tesla P100 GPU, 13 GB memory. Due to the limited time and resources, we narrowed down the Places365(1.8

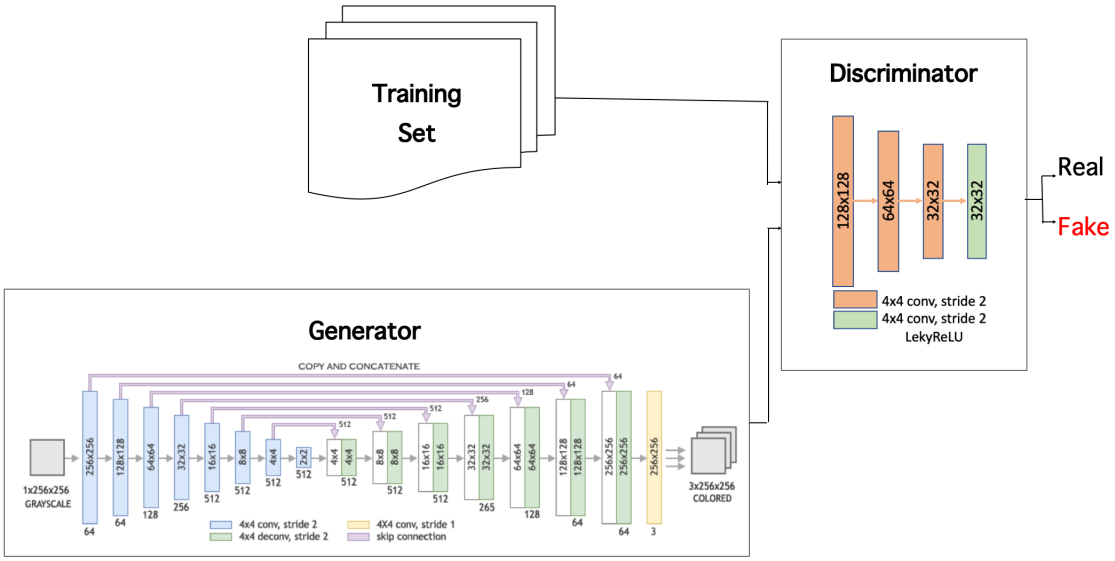

Figure 9: The conditional DCGAN network architecture [7]

million train images) dataset and formed our dataset: subset of Places365 which is mainly focused on outdoor scenes, like forest, parks, gardens, and etc.

Observations and Training Strategies: We first trained the original conditional DCGAN model for different epochs, 10, 20, 30 ... 50. We found that at some epoch around 20, the model encountered the collapse problem and the generator started to colorize images with some similar strategies, for example, filling extremely chromatic colors to the middle of each image.

To solve this problem, we changed the loss function of generator to MSE (mean of square error) of generated and source images from MAE (mean of the absolute error), and adjusted our approaches when trainning.

Here are several observations and strategies we have in our expriments:

- The number of epochs for training GAN matters: Since after some epochs, the network collapse.
- Alternative cost function can work
- Save several training checkpoints: Since the unstable training features of GAN, it is better to store some checkpoints for restoring and resuming training.
- Adjust learning rate when loss function started to plateau.
- Set an appropriate batch normalization.

2.2.3 Experimental Results and Discussion

Fig. 10 shows some results on epochs 10,20,30 of original conditional DCGAN model. We can see that the original conditional DCGAN does not have a robust performance during the training. When the epoch reaches 10, it can not give multiple colors to different objects well, while when epoch reaches 20 or more, it over-colorize the images and often has some obvious errors. It seems to follow the common disadvantages

Figure 10: Results on original conditional DCGAN model

of GANs, which collapses at some epoch. However, after changing the loss function of generator and adjusting the learning rate, our improved model seems have a better

outcome as shown in Fig. 11 which shows some results on epochs 15,18,21 of improved model. We also have some experimental strategies sharing, like the alternative loss function alternation, learning rates adjustment and checkpoints saving as listed in the above subsection.

Figure 11: Results on improved conditional DCGAN model

3 Conclusions

In this project, we first investigated some possible solutions to the this problem and had a comprehensive understanding of the background and approaches. Then, we followed the design and architecture of two models, CNN and conditional DCGAN, to perform the experiments on a subset of Places365. At last, we improved the existing models and had comparisons and discussions on our results.

We expected DCGAN would perform much better than CNN model at the very start. However, the experiments and results of CNN and DCGAN models show that GANs may not always perform better than some traditional approaches, and it might not always be a solution to every problem.

References

[1] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, 35(4):110, 2016.

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[5] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016.

[6] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[7] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization using generative adversarial networks. In *International Conference on Articulated Motion and Deformable Objects*, pages 85–94. Springer, 2018.

[8] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[10] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[11] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016.