Image Colorization with Generative Adversarial Networks

Student Name: Boyan Li, Yichen Li, Lishuang Wang

Student ID: 11912914, 11912433, 11911012

1. Introduction

Image colorization assigns a color to each pixel of a target grayscale image. It is a classical problem in computer visual, which need to use a grayscale photograph as input, and expected output a a plausible color version of the photograph. This problem is somewhat challenging because it is multimodal -- a single grayscale image may correspond to many reasonable color images.

2. Related works

Colorization methods can be roughly divided into two categories: scribble-based colorization [1], [2], [3], [4], [5] and example-based colorization. The scribble-based methods typically require substantial efforts from the user to provide considerable scribbles on the target grayscale images. It is thus time-assuming to colorize a grayscale image with fine-scale structures.

Scribble-based colorization Levin et al. [2] propose an effective approach that requires the user to provide colorful scribbles on the grayscale target image. The color information on the scribbles are then propagated to the rest of the target image using least-square optimization. Huang et al. [1] develop an adaptive edge detection algorithm to reduce the color bleeding artifact around the region boundaries. Yatziv et al. [5] colorize the pixels using a weighted combination of user scribbles. Qu et al. [4] and Luan et al. [3] utilize the texture feature to reduce the amount of required scribbles.

Example-based colorization Unlike scribble-based colorization methods, the example-based methods transfer the color information from a reference image to the target grayscale image. The example-based colorization methods can be further divided into two categories according to the source of reference images: Colorization using user-supplied example(s) and Colorization using web-supplied example(s).

Our work is based on the paper: Image Colorization with Generative Adversarial Networks.

3. Method

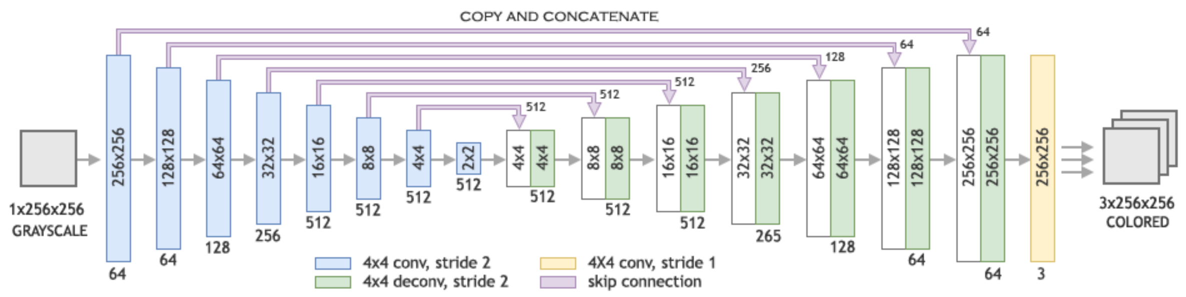
Generative adversarial networks (GANs) is a type of generative model. A GAN is composed of two smaller networks called the generator and discriminator. The generator's task is to produce results that are indistinguishable from real data. The discriminator's task is to classify whether a sample came from the generator's model distribution or the original data distribution. Both of these subnetworks are trained simultaneously until the generator is able to consistently produce results that the discriminator cannot classify.

Conditional generative adversarial networks (conditional GAN) is a kind of GAN to address the problem that the input of the generator is randomly generated noise data *z* in a traditional GAN, which is not applicable to the automatic colorization problem because grayscale images serve as the inputs of our problem rather than noise.

In our baseline model, we use **U-Net** to find a direct mapping from the grayscale image space to color image space. The architecture of the model is symmetric, with *n* encoding units and *n* decoding units. The contracting path consists of 4 × 4 convolution layers with stride 2 for downsampling, each followed by batch normalization and Leaky-ReLU activation function with the slope of 0.2. The number of channels are doubled after each step. Each unit in the expansive path consists of a 4 × 4 transposed convolutional layer with stride 2 for upsampling, concatenation with the activation map of the mirroring layer in the contracting path, followed by batch normalization and ReLU activation function. The last layer of the network is a 1 × 1 convolution which is equivalent to cross-channel parametric pooling layer. We use tanh function for the last layer. The number of channels in the output layer is 3 with L*ab** color space. We train the baseline model to minimize the Euclidean distance between predicted and ground truth averaged over all pixels:

$$J(x;\theta) = \frac{1}{3n} \sum_{\ell=1}^{3} \sum_{p=1}^{n} \| h(x;\theta)^{(p,\ell)} - y^{(p,\ell)} \|_2^2$$

where *x* is our grayscale input image, *y* is the corresponding color image, *p* and **l** are indices of pixels and color channels respectively, *n* is the total number of pixels, and *h* is a function mapping from grayscale to color images.



We use Deep Convolutional GANs (DCGAN) for the generator and discriminator models and the architecture was also modified as a conditional GAN instead of a traditional DCGAN. The architecture of generator *G* is the same as the baseline model. For discriminator **D**, we use similar architecture 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. All convolution layers are followed by batch normalization, leaky ReLU activation with slope 0.2. After the last layer, a convolution is applied to map to a 1 dimensional output, followed by a sigmoid function to return a probability value of the input being real or fake. The input of the discriminator is a colored image either coming from the generator or true labels, concatenated with the grayscale image.

4. Experiments

4.1 Datasets

We use tiny-imagenet-200, which can be obtain by Tiny ImageNet. This dataset contains 200 classes, each class include 500 images in train set, 50 in validation set and 50 in test set. The images in the dataset are colorful, with size 64×64.

4.2 Implementation Details

For training our network, we used Adam [15] optimization and weight initialization as proposed by [16]. We used initial learning rate of 2 × 10−4 for both generator and discriminator and manually decayed the learning rate by a factor of 10 whenever the loss function started to plateau. For the hyper-parameter *λ* we followed the protocol from [5] and chose *λ* = 100, which forces the generator to produce images similar to ground truth.

GANs have been known to be very difficult to train as it requires fifinding a Nash equilibrium of a non-convex game with continuous, high dimensional parameters. We followed a set of constraints and techniques to encourage convergence of our convolutional GAN and make it stable to train:

1. **One Sided Label Smoothing**

   Deep neural networks normally tend to produce extremely confifident outputs when used in classifification. It is shown that replacing the 0 and 1 targets for a classifier with smoothed values, like .1 and .9 is an excellent regularizer for convolutional networks. In this technique we smooth *only* the positive labels to 0.9, leaving negative labels set to 0.

2. **Batch Normalization**

   Batch normalization is proven to be essential to train both networks preventing the generator from collapsing all samples to a single point. Batch-Norm is not applied on the first layer of generator and discriminator and the last layer of the generator.

3. **All Convolutional Net**

   Strided convolutions are used instead of spatial pooling functions. This effectively allows the model to learn its own downsampling/upsampling rather than relying on a fifixed downsampling/upsampling method.

4. **LeakyReLU Activation Function**

   Radford et al. [6] showed that using leaky ReLU activation functions inthe discriminator resulted in better performance over using regular ReLUs.

The loss in training is designed as follows:

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

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

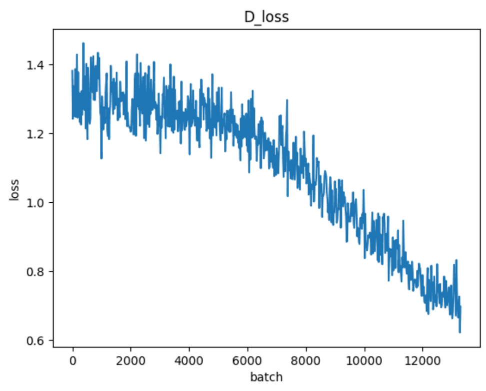
4.3 Metrics

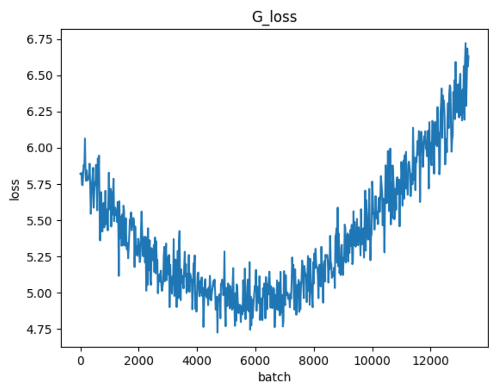
We used the average L1 loss of generated and real images as a performance measure. During training, we recorded the loss and plotted the change image. Moreover, we recorded Discriminator loss and Generator loss as well.

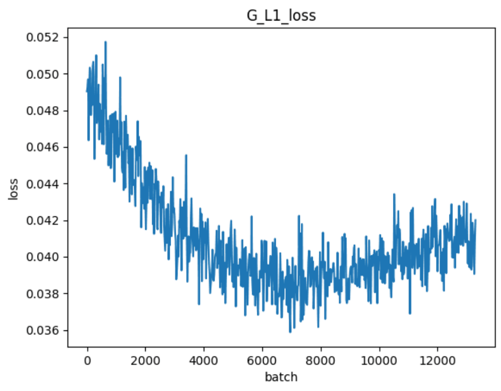
4.4 Experimental design & results

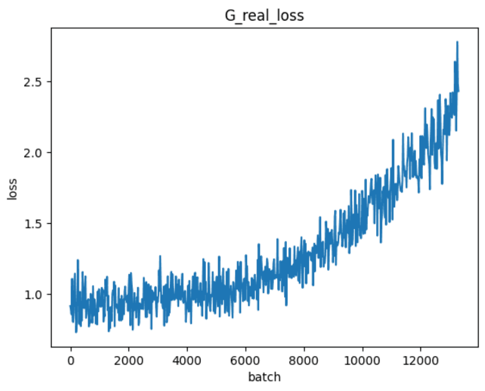
In this project, we record the progress of four different loss descending. Additionally, we convert some gray images to colorful images to show the algorithm work. Finally, we compare the colorization performance of different iterations of the model. The following are some results.

In the following four pictures, we can see the loss descending of the generator and the discriminator.



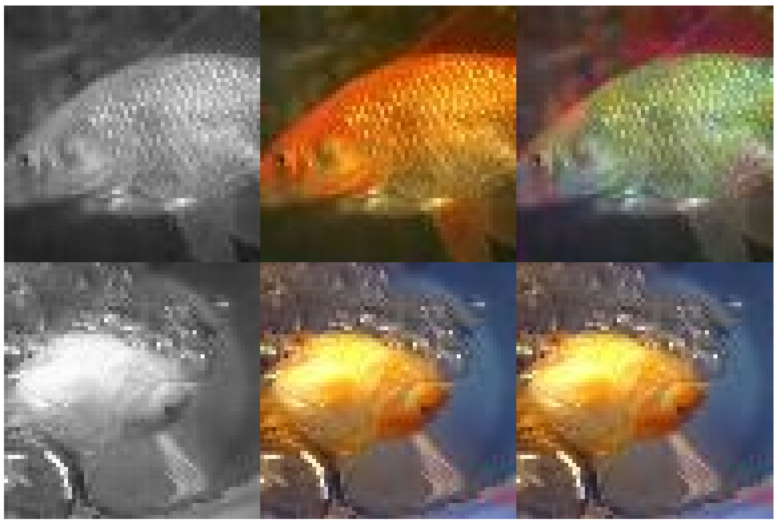






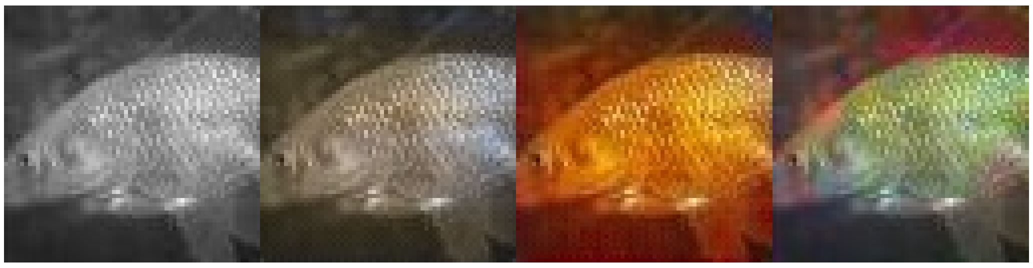
Owning to our training progress is two neural network against, the loss of each network are not simply decrease.

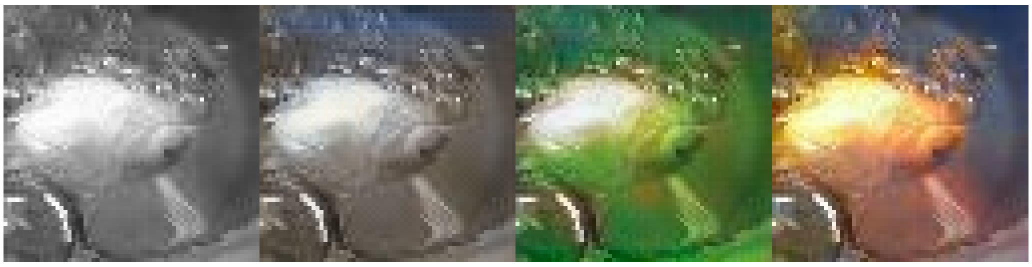
In the following images, we show the colorization ability of our model.



The left picture are the gray images, and the middle image are the colorful images we colored, the right images are the origin images.

In the following images, we show the increasing colorization ability of our model.





From left to right, the second images are generated from the initial iteration of the model, the third are from the 15th iteration, the last images are from the 54th iteration. We can find the ability of our model is increasing.

5. Conclusion

In this project, we learned the principle and variation of generative adversace model in deep learning, using U-Net network based on Encoder and Decoder architecture as the skeleton of generator. The main challenges were to reproduce the model using Pytorch and to train it, as the GAN model training was often unstable. Using a few tricks such as label smoothing, batch normalization, and LeakyRelu activation functions, we eventually overcame these difficulties and completed the grayscale image coloring task to some extent.

Contributions

- Boyan Li: 33.3% (Coding the Generator and Discriminator)
- Yichen Li: 33.3% (Coding the Dataset and Utils)
- Lishuang Wang: 33.3% (Report)

Reference

1. Y.-C. Huang, Y.-S. Tung, J.-C. Chen, S.-W. Wang, and J.-L. Wu, “An adaptive edge detection based colorization algorithm and its applications,” in Proceedings of the 13th Annual ACM International Conference on Multimedia, ser. MULTIMEDIA ’05, 2005, pp. 351–354. ↩ ↩

2. A. Levin, D. Lischinski, and Y. Weiss, “Colorization using optimization,” in ACM SIGGRAPH 2004 Papers, 2004, pp. 689–694. ↩ ↩

3. Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, “Natural image colorization,” in Proceedings of the 18th Eurographics Conference on Rendering Techniques, ser. EGSR’07, 2007, pp. 309–320. ↩ ↩

4. Y. Qu, T.-T. Wong, and P.-A. Heng, “Manga colorization,” in ACM SIGGRAPH 2006 Papers, ser. SIGGRAPH ’06, 2006, pp. 1214–1220. ↩ ↩

5. L. Yatziv and G. Sapiro, “Fast image and video colorization using chrominance blending,” Trans. Img. Proc., vol. 15, no. 5, pp. 1120– 1129, 2006. ↩ ↩

6. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015. ↩