Image Colorization with Generative Adversarial Networks

Andrew Ehrenberg
University of California - Los Angeles
andyehrenberg@ucla.edu

Abstract

Various Generative Adversarial Network architectures are compared in their performance of realistically adding color to greyscale images. Challenges regarding training these and future techniques for improvement are discussed.

1. Introduction

Image Colorization can be posed as the problem of predicting two channels from one channel representing a greyscale image. A naive approach would assume needing to predict three new channels (RGB) from a greyscale image, but the CIELAB (usually referred to as 'Lab') colorspace uses the same information for a greyscale image as its first channel, and the other two control color. Convolutional Neural Networks (CNNs) with specialized loss functions [1] have been shown to perform well at this task, but other research has suggested that Generative Adversarial Networks (GANs) [2] generalize the task more effectively [3]. I will look at the results of three GAN architectures trained on the CIFAR-10 and Places 365 [4] data sets and compare their efficacy. These architectures vary in their complexity and techniques for normalization.

1.1. ResNet Generator with PatchGAN discriminator

In this architecture, the generator is a ResNet [5] that takes in images with one channel, and outputs two channel images of the same size as the input. Between convolutional layers, batch normalization followed by ReLU activation is performed. The discriminator is also fully convolutional, classifies whether each NxN region of an image is real or fake, and takes the average of these classifications to make its decision whether the entire image has been generated, or is real [6]. Between layers, spectral normalization, and Leaky ReLU activation with slope of 0.2 is performed, as using this activation in the discriminator was shown by Radford et al. [7] to help performance. The use of spectral normalization has been shown to stabilize the training of GANs by Miyato et al. [8], and produce better results than other regularization techniques such as weight clipping and gradient penalty. The final layer of the discriminator uses Tanh activation, as suggested by [9]. Convolutional layers are initialized using Uniform Xavier Initialization [10]. The architecture has over 10 million parameters, meaning that it will be difficult to train given my resources., especially given that it is to be trained on 256x256 images from the Places 365 dataset. This model is largely inspired by the work of of Isola et al., and their research's GitHub repository [11] provides a good API for training this model. Most alterations to code were made in simplifying the model, weight initialization, and use of spectral normalization.

1.2. U-Net Generator with 1x1 PatchGAN discriminator and Spectral Normalization

In the interest of generating more acceptable results with constrained time and computational power, the following models contain fewer parameters, and are trained on lower resolution images from the CIFAR-10 dataset. The first smaller architecture uses a U-Net with 10 layers (4 downsampling, 4 upsampling). U-Nets are advisable for this task because they allow information to be processed at low-level representations while still carrying over high-dimensional information with their skip connections [12]. The downsampling is achieved by using 4x4 convolutions with strides of 2, as opposed to max or average pooling. Each convolution is followed by batch normalization and Leaky-ReLU activation. Transpose convolutions are followed by batch normalization and ReLU activation. The two channel output of the U-Net is concatenated with the greyscale input, and fed into a 1x1 (pixel-wise) PatchGAN discriminator with no normalization followed by Leaky-ReLU activation in the first layer, spectral normalization and Leaky-ReLU activation in the middle three layers, and no normalization with Tanh activation in the final layer.

1.3. U-Net Generator with 1x1 PatchGAN discriminator and Batch Normalization

This architecture is the same as that of section 1.2, except the middle three layer of the discriminator use batch normalization instead of spectral normalization. This allows for comparison between the performance of these normalization techniques. The Models for 1.2 and 1.3 are heavily inspired by the work of Károly Harsányi, and his GitHub repository [13] provides a good API for training these models and outputting results. Slight code alterations were made to the networks and scripts for training and testing. The generator and discriminator networks in models 1.1 through 1.3 were all trained with the Adam optimizer.

2. Results

The architecture from 1.1 proved to be too complex to effectively train on a CPU. After 24 hours of training, not even 1 epoch was completed, and sample generated images were not very convincing, usually with low saturation and appearing more as a sepia version of the grayscale image. A few examples are given in Figure 2.1.

The other architectures had more impressive results given the constraints of this project. After 12 hours of training for each model, more vibrant colors were seen, indicating the these models learned much more quickly than the model with more parameters. The U-Net architecture that used spectral normalization usually produced more realistic images at the cost of occasionally being less saturated, whereas the one that used batch-normalization had more vibrant generated images, but they were less believable. Examples of images produced by the two U-Net models can be seen in Figure 2.2.

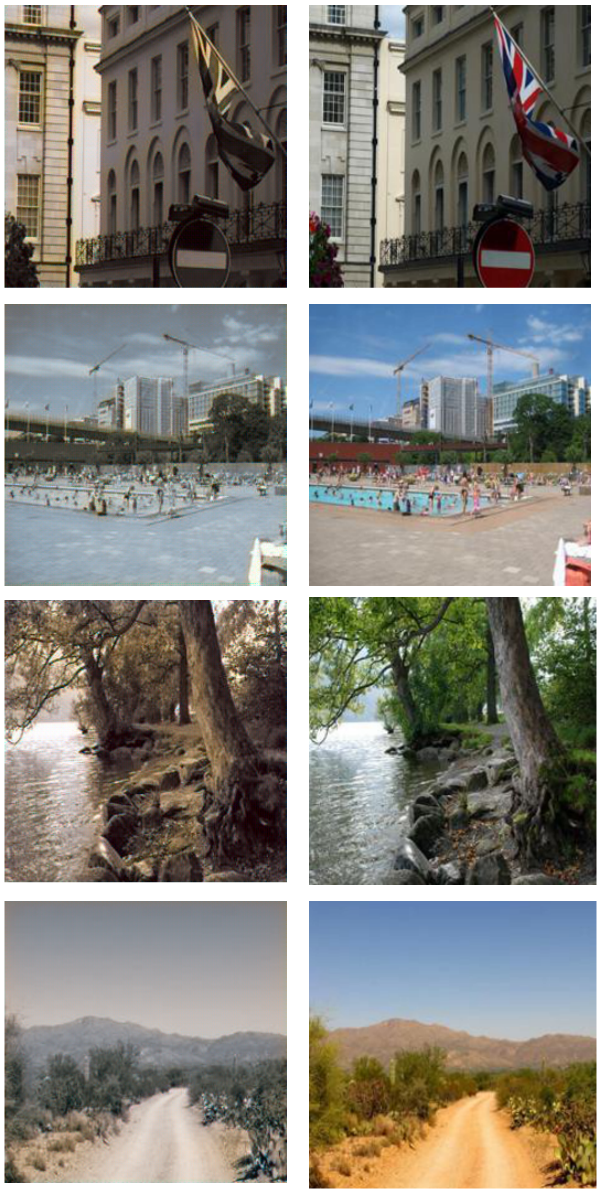

Figure 2.1: Samples from Model 1.1. Colorized images on left, real images on right

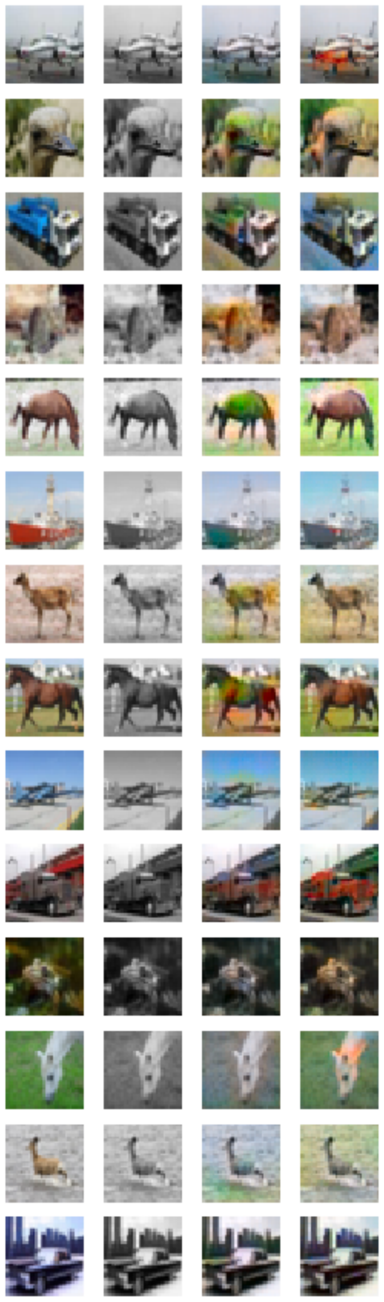

Figure 2.2: Samples from Models 1.2 and 1.3. From left to right: Real images, grey images, images from network with batch normalization, images from network with spectral normalization

3. Discussion

Of the three architectures, the U-Net with spectral normalization in its discriminator produced the most believable results. This suggests that the spectral normalization stabilized the training procedure as intended. The model with batch normalization in the discriminator did do better at generating vivid colors, especially in situations where the subject of the image could have various plausible colors, such as pictures of cars. Further work should include training the larger model for longer on a GPU to see if it gets good results. Additionally, training and testing on larger images will show whether these models can scale at higher resolutions. Exploration of generalizing these models to other tasks besides colorization, such as converting satellite imagery into maps, image segmentation, and augmentation of audio signals could result in fruitful findings as well.

4. References

1. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. ECCV. 2016.
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. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. 2017.
4. Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. 2016.
5. He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.
6. Isola, Phillip and Zhu, Jun-Yan and Zhou, Tinghui and Efros, Alexei A. Image-to-Image Translation with Conditional Adversarial Networks. 2017.
7. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.
8. Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. 2018.
9. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.
10. Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 2010.
11. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation in PyTorch, 2019. GitHub repository: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
12. O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, 2015.
13. Károly Harsányi. Image Colorizing with Generative Adversarial Networks on the CIFAR10 Dataset, 2019. GitHub repository: https://github.com/karoly-hars/GAN_image_colorizing

Losses After 10 Epochs

	Model 1	Model 2	Model 3
Gen adv loss	NA	0.8640	0.8323
Gen L1 loss	NA	0.0825	0.0812
Disc loss	NA	0.5747	0.5758
Disc acc	NA	0.7320	0.7367

In the first architecture, the generator is a ResNet that takes in images with one channel, and outputs two channel images of the same size as the input. Between convolutional layers, batch normalization followed by ReLU activation is performed. The discriminator is also fully convolutional, classifies whether each NxN region of an image is real or fake, and takes the average of these classifications to make its decision whether the entire image has been generated, or is real. Between layers, spectral normalization, and Leaky ReLU activation with slope of 0.2 is performed. The final layer of the discriminator uses Tanh activation. Convolutional layers are initialized using Uniform Xavier Initialization.

The first smaller architecture uses a U-Net with 10 layers (4 downsampling, 4 upsampling). Before being fed into the U-Net, the image is resized to a 35x35 image. The downsampling is achieved by using 4x4 convolutions with strides of 2, as opposed to max or average pooling. Each convolution is followed by batch normalization and Leaky-ReLU activation. Transpose convolutions are followed by batch normalization and ReLU activation. The two channel, 35x35 output of the U-Net is concatenated with the greyscale input, and fed into a 1x1 (pixel-wise) PatchGAN discriminator with no normalization followed by Leaky-ReLU activation in the first layer, spectral normalization and Leaky-ReLU activation in the middle three layers, and no normalization with Tanh activation in the final layer.

The second smaller architecture is the same, except it uses batch normalization instead of spectral normalization in the discriminator.