Automatic Image Colorization Using Conditional Generative Adversarial Networks

Mariya Kazachkova[1], Sarah Sukardi[1], Hugh Han[1]

1 Department of Computer Science, Johns Hopkins University, Baltimore, Maryland, United States

Abstract

In this paper, we investigate the use of conditional generative adversarial networks (cGANs) used for the task of automatic image colorization; that is the task of turning a grayscale image into a false color image that is similar to a true color image. Previously, approaches to image colorization were either done manually by a human, or resulted in images that were desaturated. We present an approach to create fully automatic image colorization, exploring several model inputs in order to determine the features that the cGAN is learning in the process of colorization and investigate cGAN model interpretability. Finally, we conclude with a qualitative and quantitative analysis of our results, as well as further directions in which this investigation can go.

I. Problem Statement

In this project, we attempt to find a method of coloring grayscale images automatically by training conditional generative adversarial networks (cGANs). cGANs are a specific class of general adversarial networks (GANs), which consist of a generator and a discriminator neural network. The generator receives noise vectors, produces images that closely resemble the actual images, and attempts to fool the discriminator into incorrectly classifying these fake images as actual images. The conditional GAN architecture was pioneered by Mirza et. al [1].

Here, the generator and discriminator neural networks play a zero-sum game. The objective of the generator is the minimize the likelihood that the discriminator correctly classifying the image, whereas the objective of the discriminator is to maximize the likelihood that the discriminator correctly classifies the image. We arrive at the following optimization problem.

$$\begin{aligned}&\min_g\{\max_d\{\mathbb{E}[\log(D(x;\theta_d))]+\\&\mathbb{E}[\log(1-D(G(z;\theta_g);\theta_d))]\}\}\end{aligned} \tag{1}$$

Here, x is sampled from the original data distribution from the dataset, and z is sampled from the data distribution of the noise. θ_d and θ_g are the weights corresponding to the discriminator and the generator, respectively. $D(\cdot;\theta_d)$ is the likelihood of correctly classifying an image, using weights θ_d, and $G(\cdot,\theta_g)$ is an image generated using weights θ_g.

cGANs are GANs that simply condition on a particular event. So using equation (1), an example of the corresponding cGAN could be something such as what follows.

$$\begin{aligned}&\min_g\{\max_d\{\mathbb{E}[\log(D(x|y;\theta_d))]+\\&\mathbb{E}[\log(1-D(G(z|y;\theta_g);\theta_d))]\}\}\end{aligned} \tag{2}$$

Note that the only difference here is that D and G are now conditioned on some event y, as $D(x;\theta_d)$ and $G(z;\theta_g)$ become $D(x|y;\theta_d)$ and $G(z|y;\theta_g)$, respectively. In our problem, the event y can be thought of as the event that the input passed through the models are black-and-white.

To narrow down on the general problem of coloring black-and-white images, we decided to focus particularly on coloring black-and-white images differently than their original coloring, but in a particular way. That is, we wanted to see how cGANs would perform when training on one particular type of dataset, and then using it to color images from another dataset. To train and test the efficacy of the cGAN, we trained and tested on 250 images from the Nature Scene Collection dataset [7], a nature-centric dataset with 9 different sets. We believed that this dataset a large enough amount of images that we thought we could produce compelling results. We then took the project further by testing whether the cGAN learns to color primarily on pixel intensity or whether it realizes the concept of an "object" when colorizing a particular object. To test this in particular, we used 250 images of brown bears taken from Google Images in order to train the cGAN, and then used images of polar bears taken from Google Images, processed them into grayscale coloring, and ran them through the trained cGAN to see if they would be colored brown. That is, we additionally wanted to see if cGANs could recognize structure within images, even if the test data and training data are obtained from two different types of datasets.

II. Methods

Originally, it was thought that variational autoencoders (VAEs) [4] could potentially be a strong method of tackling the problem of generating colored images from black-and-white images, and we thus explored and prototyped methods of implementing colorization using autoencoders. However, because VAEs are probabilistic graphical models with the explicit goal of latent modeling, it was decided that cGANs would likely perform better, as they are designed explicitly to optimize for generative tasks given some condition (such as a black-and-white image). That is, while VAEs can be used for generative purposes, they are more ideal in problem sets where latent representations are more important, as VAEs are designed to learn latent representations. Hence, it was decided to tackle the problem using cGANs, which are explicitly optimized for generative tasks.

In order to obtain our initial dataset for training the cGAN architecture we ultimately decided upon, and eventually determine the cGAN architecture's efficacy on general images, we downloaded the Nature Scene Collection Dataset [7]. We resized the images into 256x256 square images to have a consistent image input into the cGAN. We then implemented a data pipeline which automatically obtained a grayscale copy of the image that would then be input into the cGAN.

For our second experiment, we obtained data from Google images on brown bears and white bears. The following `google-images-download` Github repository was used in order to scrape and download large quantities of images of brown bears from Google Images. The top 250 images of the "brown bear" search result from Google Images' search results were used as training data, and the top 10 images of "polar bear" from Google Images' search results were used as test data to evaluate our cGAN architecture. Additionally, these images were integrated into our same modular pipeline that resized and grayscaled the images.

In addition to investigating structural similarity, we decided to perform these two experiments in order to learn more of the nature of how the cGAN learns, and whether it truly learns larger structures or rather makes decisions based on more localized regions.

In order to train the cGAN, we began with starter code that established a base cGAN architecture from the `pix2pix` Github repository. After initializing the cGAN provided by `pix2pix` onto Google Cloud, the model was trained for 50 epochs.

III. Results

The computational resources that we used to train and test our cGAN models were several instances of the Google Cloud Platform engine. We initially had trouble getting CUDA to be recognized by our version of PyTorch, but after setting up several instances in which we were able to get CUDA and CuDNN running on the cloud machine, we trained our model on 2 NVIDIA k80 GPUs with 4 GB of memory for 2 hours.

We first trained the cGAN using the Places dataset from CSAIL MIT on a relatively small subset of 250 images of natural places for 50 epochs. The size of our test set that we used to evaluate the efficacy of our model was 30 images. From training on 250 images, we were already able to obtain auto-colorized results that were very convincing.

Training on more epochs led to considerably better results. After training on 5 epochs, the output of the colorization procedure was poor, with low contrast images that were all tinted one color (similar to how a grayscale image is all "tinted" one color). However, after training on 50 epochs, our results considerably improved, with natural backgrounds being colorized correctly the majority of the time. See below for several example of images output by our cGAN, along with the original image output:

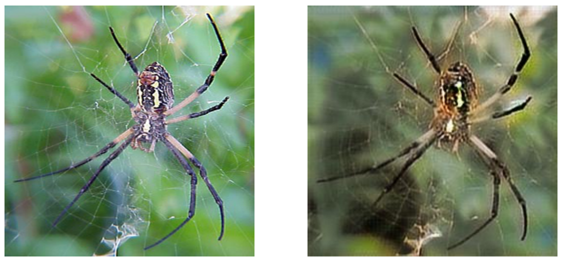

Figure 1 *Example of Original Image Juxtaposed with Colorized Image*

One can see from the above figure that the output colorized image is remarkably similar to the original color image. The cGAN has learned to color the background green and the body of the spider to be brown, and the boundary between the color of the legs of the spider and the background is very fine, indicating high accuracy of colorization and that the training was completed successfully. Additionally, the SSIM, or structural similarity, of the above image with the original image is 0.5809.

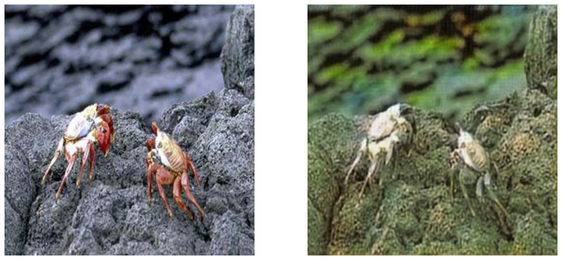

Figure 2 *Another Example of Original Image Juxtaposed with Colorized Image*

However, training on only 250 images resulted in some limitations regarding colorization of test images that were not representative of the training dataset. Figure 2 is an example of an image which did not colorize well. One can see that the background and rocks are colorized to be green. We posit this is because our training dataset had a large amount of images with trees, grass, and green foliage. Thus, the cGAN interpreted the background, which has a mottled texture, to be green foliage rather than black rocks. Additionally, the crabs are not colored orange, as they should be. This is because our small training dataset of 250 images did not contain any images of crabs. Since orange is not a color often found in nature, the cGAN was predictably unable to colorize the crabs correctly when it did not train on any crabs. Finally, the SSIM of the above colorized image with the original image was much lower than that of figure 1, which was the well-colorized image, and the value was 0.4663. See below the loss function of running the cGAN for 50 epochs. Especially notable about the loss function is how the shape of the two lines (orange being the generator, and blue being the discriminator) oscillates. This coheres with our conception of how a cGAN is supposed to work, as the generator and discriminator networks play a zero-sum game, where the objective of the generator is to minimize the likelihood of the discriminator classifying the image, and the objective of the discriminator is to maximize its own likelihood at classifying the image. One can see from the loss function that when the loss function for the generator goes down, that of the discriminator goes up, indicating that as one improves, the other "compensates" for it. However, the total slope of both lines is decreasing, indicating that learning is occurring. Our loss function, thus indicates that our cGAN is implemented correctly and also learning.

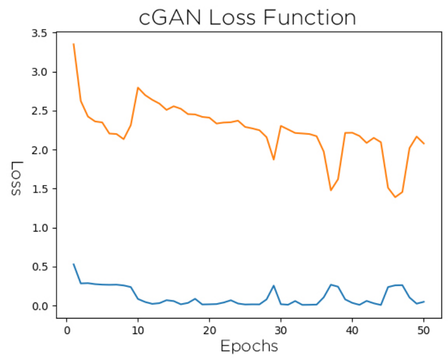

Figure 3: *Generator and Discriminator Loss as a Function of No. of Epochs*

Our next experiment to evaluate the performance and capabilities of the cGAN was an experiment to determine the kinds of features the cGAN was learning. After training the cGAN exclusively on brown bears for 50 epochs, we tested it on 10 images of polar bears. We retrieved the following results from the testing procedure:

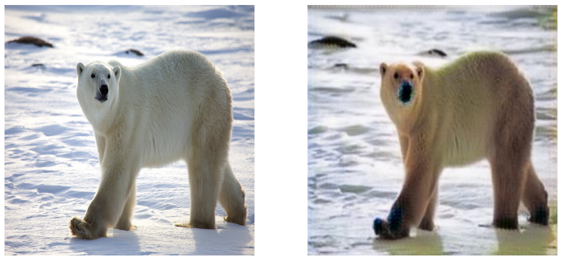

Figure 3 *Example of Original Image Juxtaposed with Colorized Image of a Polar Bear*

One can see from the sample image that where the snow in the background is almost perfectly correctly preserved as being a white color, as it should be, the color of the bear is much darker, similar to the brown bears that were trained upon. However, if one were to sample the pixel values there, they would be much greater relatively compared to the pixel values of a brown bear. Thus, the fact that the bear is colorized to be a brown color (like the brown bears it was trained on) rather than white (how one would predict it would be colored based on pixel intensity) indicates that the cGAN is in fact learning structure, rather than just making decisions on more localized regions.

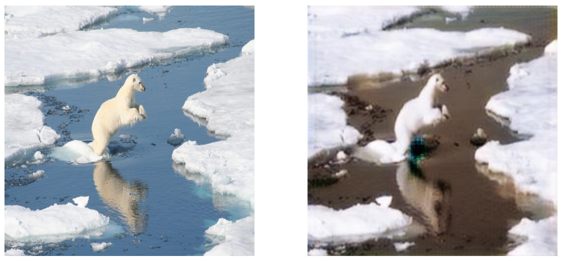

Figure 4 *Another Example of Original Image Juxtaposed with Colorized Image of a Polar Bear*

It is also interesting to note that in Figure 4 we see that instead of coloring the polar bear brown, the cGAN produced an image with the water colored brown. After looking again at our dataset of brown bear images, we noticed that almost all of the images contained a brown bear in the center of the image with a lighter background. The bear was often the largest thing in the image (in terms of literal space taken up in the image). Thus, it could be the case that the cGAN was learning to color large objects in the center of images that were on a lighter background brown, ultimately producing the brown water we see in Figure 4. We thought that this was a very interesting result, and would have wanted to try and reproduce this phenomenon with other datasets if we had more time. Overall, it seems to be the case that understanding exactly what the cGAN is learning is very difficult.

IV. Discussion

From our results, we can conclude that the cGAN is extremely effective at image colorization, even when constrained to a small dataset of images for a limited number of epochs.

Some limitations that we encountered included the size of the dataset as well as the computing resources we had. We had $50 of credit to use on training, which limited the amount of time that we could train for. Additionally, we only trained on 250 images for both experiments with the Places dataset, as well as our custom brown bear dataset. This is because our cloud instance had a very limited amount of memory, as many programs had to be installed onto the instance, which led to only a small amount of memory able to be allocated for storing the image files which were quite large, as the images were high resolution. Thus, in the future, to properly evaluate how powerful the GAN actually is at image colorization, we would train on a much larger dataset of images ranging in the tens of thousands to possibly even the hundreds of thousands when we have more computing resources. Additionally, we would also use a more diverse range of images in our dataset, expanding it from just places to a more wide-ranging dataset like ImageNet which also includes photos of objects and people. This would be in order to see how the cGAN performed with a more difficult set of conditions and search space.

Several things that we learned about cGANs were that they are remarkably robust, working well even for very small datasets and relatively short training times (in literature and standard practice, cGANs are trained for 200 epochs. Comparatively, we trained our cGAN network for 50 epochs). We also learned that cGANs differ from typical image-to-image formulations in that they learn a structured loss that penalizes the joint configuration of the output. This learned loss results in a loss metric that can, in theory, penalize any possible learned structure that differs between the output image and the target image, and is thus a much more robust metric.

Some advice we would give to future students in Deep Learning would be to gain a thorough understanding of the statistical and machine learning fundamentals necessary to understanding the later concepts. Particularly, sections on Bayesian methods in statistics and understanding the fundamentals of backpropagation are particularly useful from the introductory sections. Once a thorough understanding of these has been established, the much more complicated cutting-edge concepts become much more approachable. This basis was necessary for us to understand how the cGAN architecture and generative optimization procedure worked. Additionally, we would suggest students to just have fun! This is a truly cutting-edge class and we're lucky to be able to have class at Hopkins already that teaches such a new field.

V. Appendix

References

[1] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets, 2014;

arXiv:1411.1784.

[2] Yun Cao, Zhiming Zhou, Weinan Zhang and Yong Yu. Unsupervised Diverse Colorization via Generative Adversarial Networks, 2017; arXiv:1702.06674.

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks, 2016, CVPR 2017; arXiv:1611.07004.

[4] Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed and Alexander Lerchner. Early Visual Concept Learning with Unsupervised Deep Learning, 2016; arXiv:1606.05579.

[5] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning Deep Features for Scene Recognition using Places Database. Advances in Neural Information Processing Systems (NIPS), 2014.

[6] Laurent Dinh and Vincent Dumoulin. Training neural bayesian nets. NCAP 2014 Summer School, 2014.

[7] Geisler WS Perry JS (2011). Statistics for optimal point prediction in natural images. Journal of Vision. October 19, 2011 vol. 11 no. 12 article 14.