Image Colorization using Conditional Generative Adversarial Networks

CS-9-A
Group Members:

Abdullah Bin Omer Zia – 291214

Tafheem-ul-Islam Malik – 321906

Introduction:

Colorization of images is a difficult topic to solve because of the many different imaging situations that must be handled by a single algorithm. Deep learning has been used to colorize black and white photographs, which is now an outstanding presentation for the real-world use of machine learning in our daily lives. In the perspective that there is no distinct colorization of a black and white image without any foreknowledge, this issue is ill-posed. Many artifacts can, in fact, have various colors. This is valid not only for manmade items, such as plastic goods, which can come in a variety of hues but also for natural items, like tree leaves, which can change color from spring to autumn despite not changing their shape.

Our model takes an input grayscale/black and white input image and predicts the plausible colors for it to make it look as real as possible. Through this project, we are trying to make it easier to colorize older (or newer) images available in only black and white, a process that was previously only doable by hand in programs such as Photoshop – a painstaking process.

This problem is important as it lets us visualize how some things look actually rather than seeing the black and white image of the same object which won't help us visualize it the same way the colored one does.

Literature Review:

Many different deep learning-based colorization techniques have been proposed over the last few years. Few of the papers that mention this problem are Colorful Image Colorization [1], Let there be Color! Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification [2], and Image-to-Image Translation with Conditional Adversarial Networks [3] (aka pix2pix).

The authors of Let there be Color! Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous managed the problem as a regression assignment.

The authors of the Colorful Image Colorization tackled the topic as a **classification** challenge, taking into account the problem's unpredictability (e.g., A shirt worn by a person in the image can take many distinct and legitimate colors, and we can't say for sure what color actually it will be.).

The approach we are going to follow will be the one used by *pix2pix* - two losses are used: an **L1 loss** and a **GAN loss**, described in the equations below.

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) = & \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]\end{aligned}$$

Figure 1: Conditional GAN Loss

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Figure 2: Combined Loss Function we Optimize

Lambda in the above equation is a coefficient we can use to balance the contribution of the L1 loss to our final loss function.

We chose to pretrain the generator independently in a supervised and deterministic manner in the GAN game where neither generator nor discriminator knows anything about the task at the beginning of training.

Dataset:

We had multiple options for the dataset such as COCO, ImageNet, etc. The dataset we're using is COCO unlabeled images 2017 which is royalty-free and is directly available on the "coco dataset" website. This dataset consists of approximately 123,000 colored images out of which we'd be using approximately 10,000 for our training. We used a subset of data because of less computational power to train the data on our side. It took roughly 7-8 hrs. on our RTX 2060

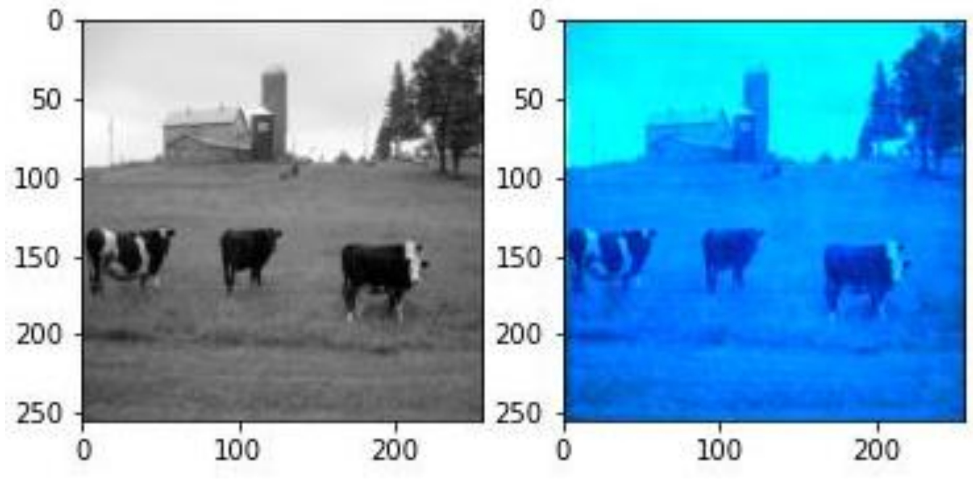
MAX-Q to train the model. We divided our data into 8000 images for training and 2000 for validation.

We used L*a*b color space for our project. As you may already know, when we load an image in our program, we obtain a rank-3 (height, width, color) array with the color data for the image on the last dimension of the rank-3 array. This color data in the RGB color space represents how much Red, Blue, and Green are present for each pixel in the image. Similarly, in the L*a*b color space, we have 3 numbers here as well, but they represent different things in this case. The L channel represents how much Lightness is there for each pixel, the *a channel encodes how much green-red and the *b channel encodes how much yellow and blue each pixel is.

In the L*a*b color space, the L channel directly gives us the grayscale part of our image without any prior conversions. We feed the L channel into our model for predictions of the *a, and *b channels and concatenate it with the input L channel to give us a colorful prediction of the grayscale image, while in RGB we'd need to convert the image to grayscale and feed the converted grayscale into the model to predict 3 channels Red, Green, and Blue. So, L*a*b color space reduced the complexity in that regard.

Baseline:

The below screenshot gives us the output of our baseline model which used ResNet18 as the backbone. The below image result is produced without any training.



Main Approach:

We built our model as a Conditional GAN and used an extra L1 loss function. In a Generative Adversarial Network, we have a generator and a discriminator. In our model, the generator takes in the input grayscale image (L channel) and predicts the other 2 channels (*a, and *b) of our image. The discriminator takes in these 2 predicted channels and combines them with the input grayscale channel and tells if the image is real or fake. We keep wanting the discriminator to get better at it's predictions while simultaneously wanting our generator to produce outputs good enough to "*fool*" the discriminator. So, in the end, we want our *discriminator* loss to be as high as possible for images generated by our *generator* (as is clearly depicted in Figure 1 where the conditional GAN loss function is described).

Our **Generator** follows a **UNet architecture** based on the **ResNet-18** backbone. We built it using fastai library's built-in function and used a pretrained version of the ResNet-18 model, thereby employing transfer learning.

The below image describes the working of a UNet. It consists of convolution and max-pooling layers on the left side which downsample the image until it reaches the "bottom of the U". From there onwards, upscaling and convolution is used to increade the dimensions of the image once again (in our case to match the dimensions with the input image).

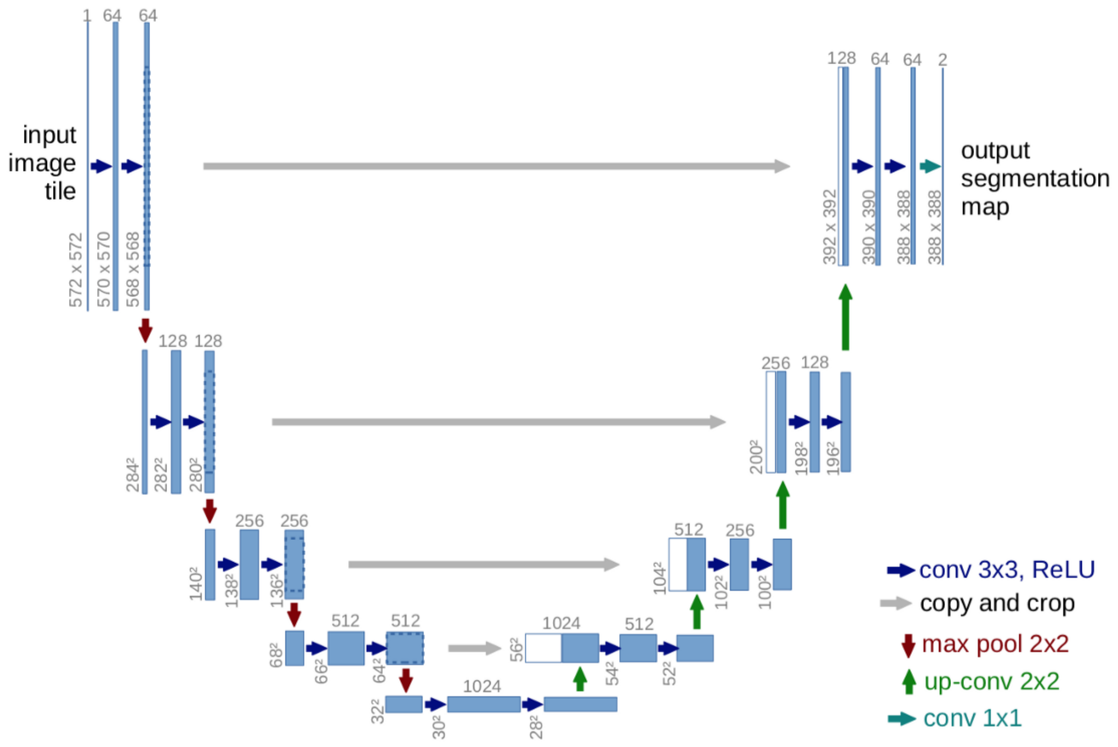
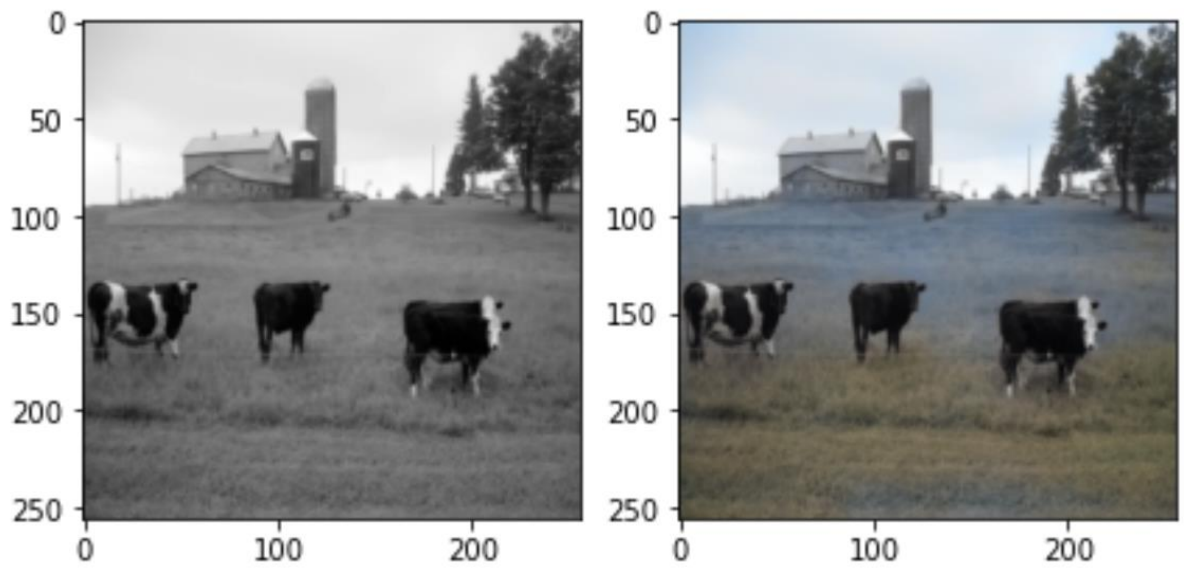


Figure 3: UNet Graphical Representation (Values are not indicative of our actual model)

Our **Discriminator** on the other hand was a Convolutional Neural Network (CNN) with Leaky-RELU as our activation function for all layers (except output) and interspersed Batch Normalization. It works as a patch discriminator with a 30x30x1 output for every image fed into it. It basically divides each image into 30*30 chunks and provides a prediction of whether each patch is real or not individually. These patches are summed and averaged to provide the overall prediction for the "*realness*" of the image.

We start by pretraining the generator using L1 loss only and we get the following results at the end of 10 epochs:

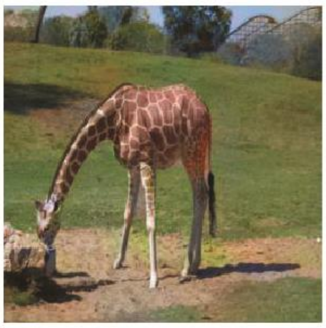


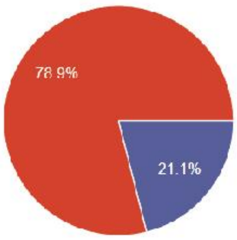
Comparing to our baseline above, we can see the model has already vastly improved and begun recognizing grass a yellowish/greenish and the sky as blueish. These results to vastly improve following our adversarial training and are shown below in our evaluation metric.

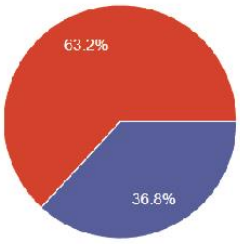
Evaluation Metric

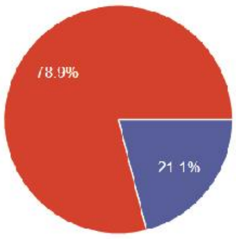
It is difficult to provide a concrete evaluation metric for the problem at hand as how an image looks can be a very subjective matter. Objective evaluation metrics (such as accuracy) will not work as well in assessing the real capability of the model for the same reason that regression alone does not work well in solving this problem. An object may have several plausible colors that may not be the same as the original image, so we let it be up to humans to decide which image looks real and which doesn't.

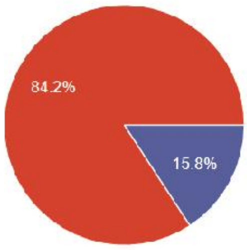
We created a Google Form to get our peers' view on how real the images output from our model look. All images were from artificially colored using our model. Here are the responses we got corresponding to each image:

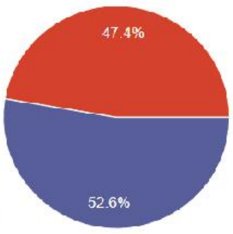


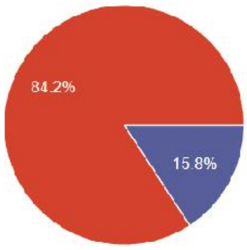


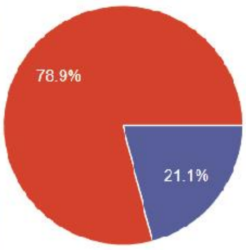


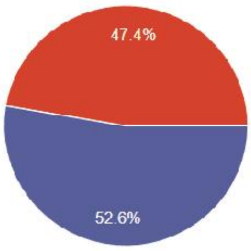












● Real
● Fake

Results & Analysis

Our model, while not perfect is showing considerably impressive results, especially on images with grass, trees and skies (commonly recurring in our dataset and not many different possible colors).

We were able to train our model for 10 epochs (pretraining the generator using L1 loss only) and 16 epochs of adversarial training, all-in-all taking us about 6 hours. Continuing to train the model for about a total of 75-100 epochs would hypothetically yield much, much better results but we did not have the computational power or resources to do so.

Future Work

Given more time and computational power, we would train the model for at least 100 epochs and using a much larger dataset (perhaps the entirety of the coco dataset with ~123k images).

Additionally, we would maybe create a Graphical User Interface for our application to allow users to more easily use our model and play around with it.

We could also try to build a model which takes in variable sized images and provides the output in the same size as the input (our model currently resizes all images to 256x256 and provides the output in the same dimensions).

---

[1] Zhang, R., Isola, P., & Efros, A. A. (2016, October). Colorful image colorization. In European conference on computer vision (pp. 649-666). Springer, Cham.

[2] 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 Transaction on Graphics (Proc. of SIGGRAPH), 35(4):110, 2016.

[3] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).