Image colorization using Generative Adversarial Networks

(Final Project)

By Simeon Kraev, Shahd Safarani and Valerie Meyer

Outline:

1. Abstract
2. Related work
3. Dataset
4. Hyperparameters
5. Network structure and design choices
6. Performance evaluation and comparison
7. Architecture, plots and layout

Abstract

The aim of the current work is to create an artificial neural network that when provided with a grayscale image as input, will be able to output a colorized version of the image. The second aim is that the colorized images should be believable colors i.e no purple grass. The approach taken here is based on (arXiv:1702.06674), using a Conditional Generative Adversarial Network (CGAN). GANs have proven themselves as a reliable method, although one that requires plenty of fine tuning. They are valuable in tasks when generation or approximation of unknown information is required. Thus the goal is create an unsupervised approach that once trained will not use user input, such as scribbles or manual color segmentation of the image. In the following chapters, the motivation and methods of our framework will be described.

Related work

Other approaches exist, whose aim is to create an unsupervised ANN with the same goal of colorization in mind. So let's examine some of the other approaches. An interesting approach that provided good results was ("Let there be color" - 10.1145/2897824.2925974), where the author's framework was based on a CNN that managed to merge global and mid-level features extracted from an image. Those features were then combined with a classification network. The colorization was then done based on features extracted from a certain class i.e outdoors, indoors. Thus they managed to avoid the sepia-toned colorization that other approaches suffered from. Another benefit was that their framework didn't take a strict resolution size as input, but rather worked with any sized images. "Image Colorization with Convolutional Neural Networks" (http://www.liangjy.com/pdf/ImageColorization.pdf) tried to reproduce their results, but did not achieve satisfying results.

"Automatic Colorization with Deep Convolutional Generative Adversarial Networks"(https://www.semanticscholar.org/paper/Automatic-Colorization-with-Deep-Convolutional-Gen-Koo/ca769bc02cb1b74a160d606fbb171afb13d0d615) framework is based as ours on Generative Adversarial Networks. The images they take as input are preprocessed the same as ours - transformed from RGB to YUV color space in addition to normalizing them. Y-channel conveys the luminance information (black and white channel), while U- and V-channels are the color or chrominance channels. The rest of their model follows GAN conventions - generated images by the generator are concatenated with the original images, and fed through the discriminator, which tries to distinguish real from fake images.

Dataset

The dataset used is one class from the LSUN dataset. Around 120 000 images of churches outdoor are used to train the network. Given that our computational resources were not high, the intuition was to use a single class. The images were then downsampled from their original resolution to 64x64 pixels for the purpose of simplifying the problem. The data has been then normalized before feeding it into the network. The last step before feeding the images into the network is to transform them from RGB color space to YUV, Y representing the grayscale images we will use for input, U and V representing the color channels or chrominance, considering the data is always kept normalized after rgb-yuv conversion. Only the Y channel is fed through the Generator, together with a matrix of a random noise and size matching the image size. Parts of the code used for the transformation to and from RGB was taken from (https://github.com/Armour/Automatic-Image-Colorization/blob/master/image_helper.py).

Network structure and design choices

The Generative Adversarial Network framework is justifiably used, based on GANs previous success in the generation of images, approximating the real distribution. In accordance with CGAN some changes were made to the standard GAN architecture. First, the input grayscale image Y is fed to every layer of the generator, to facilitate better learning throughout the network. Second, the random noise used for the initial layer is also sent to second and third layer so that perturbing noise could strengthen the approximation.
The generator consists of one ffnn to project and reshape z-vector onto image space and 6 conv-layers with stride size of 1. ReLUs are used in all conv-layers except for the last layer which uses sigmoid activation

function to output normalized data, considering we tried tanh but didn't give decent results.
The output of the generator are two feature maps that should represent the color channels UV as normalized. They are then concatenated with the original Y channel also being initially normalized and passed through the discriminator together with the original YUV image which is also initially normalized. The architecture of the Discriminator is following the conventions, with convolutional layers of increasing number of feature maps and decreasing image size. Leaky ReLUs were used in the discriminator convolutional layer. Two fully connected layers follow the discriminator without activation functions so that we get logits that could be used for calculating cross entropies.
Training was done on Amazon Web Services (AWS) by using a p2.xlarge instance. The network was trained for around 5000 steps, which represents 5 epochs.
Cross entropy was chosen as a measure for the loss function. Not to forget including batch normalization everywhere before ReLUs and Leaky ReLUs.

Hyperparameters

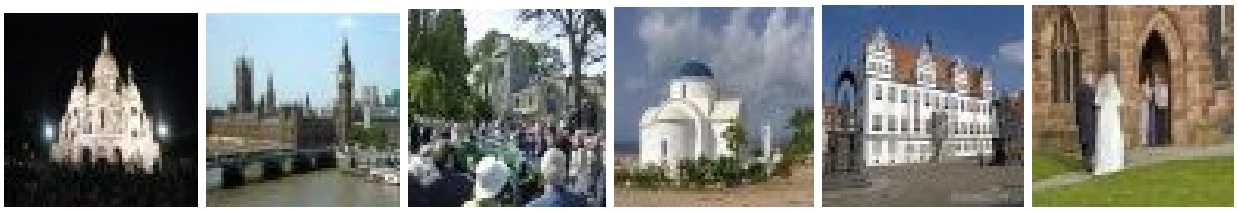
Various parameters were tested, the ones found to be the most successful are:

Different optimizers were used i.e Adam, Adagrad, RMSprop. Experimentally RMS was chosen at the end, because it showed the best results. Learning rate of 0.01 with a decay of 0.9. Batch size of 128 images and image size of 64 by 64 pixels.

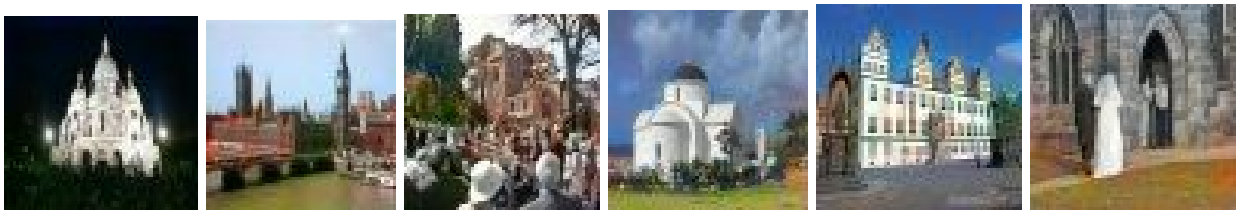
Performance evaluation and comparison

Given the subjective nature of evaluating images, evaluation was done by comparing generated and original colored images side by side.

Original Images



Predicted colors



Conclusion

Colorization using Conditional GAN (CGAN) was achieved on images of size 64x64, consisting only of a single class. The end result was decently colored images, that avoided the sepia-tone that some other approaches had. Considering our limited computational resources, we consider the project a success.

Architecture, plots and layout

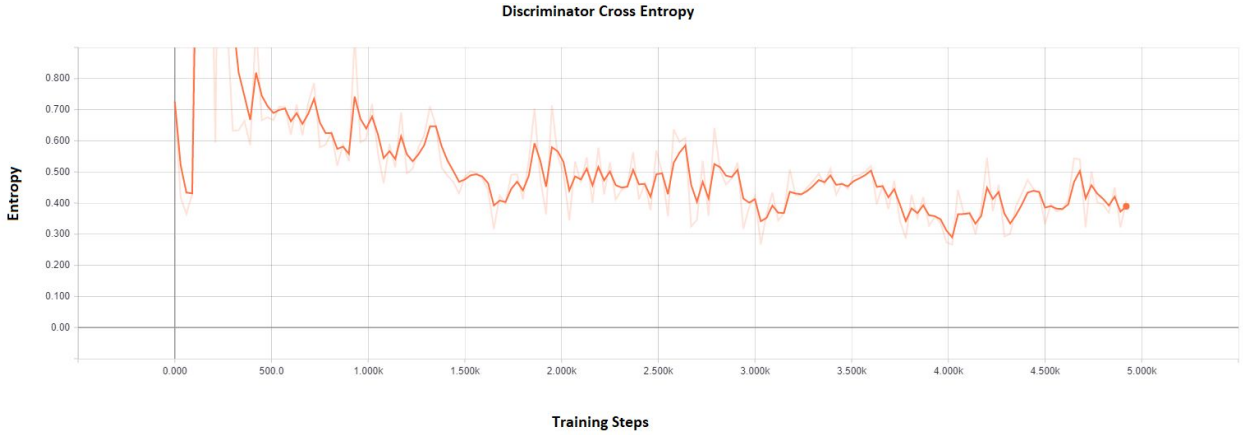
Network architecture:

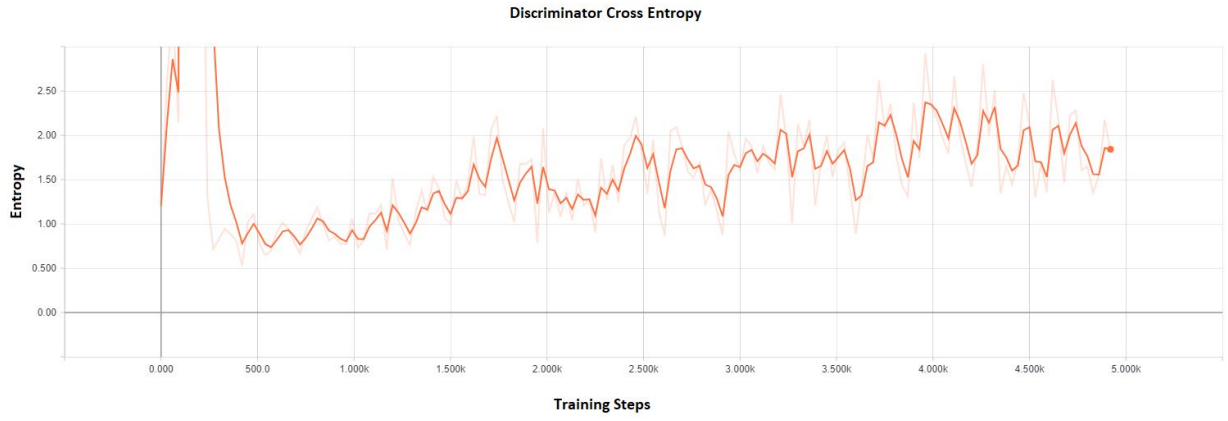
Layer - features maps, kernel size
Generator - Layer 1:128,7; Layer 2: 64, 5;Layer 3: 64, 5; Layer 4: 64, 5; Layer 5: 64, 5; Layer 6: 32, 5
stride size : 1
Activation function: ReLu, except for last conv-layer, sigmoid is used

Discriminator - Layer 1: 64, 5; Layer 2: 128, 5;Layer 3: 256, 5; Layer 4: 512, 5;
stride size : 2
Activation Function: Leaky ReLu

Plots

After 5000 training steps:





Network Graph