University of California, Irvine

EECS 195: Network Science

Colorization Utilizing Deep Learning and Conditional Random Fields

Timothy Do, Matthew Prata, Jorge Ragde, and Alex Wang

February 8th 2022

1 Introduction

In this work we address the task of Colorization using Deep Learning in addition to Colorization is the process by which colors are added to a monochrome image or film. One popular method is semantic segmentation, where the user supplies a plausible or desired color scheme to the gray-scale image in several places and the model then fills in those colors to match the partitioned image. In contrast, this project will attack the problem by producing a *plausible* color version of the photograph in a completely automated fashion. This is clearly an under-constrained problem without user intervention, but we will be embracing the uncertainty and primarily utilizing the lightness channel of the image to assign a color.

2 Approach

We will first perform a contrast enhancement method on the input image to encourage better grouping of pixels. This will be done using a fully connected CRF model [2]. We will employ the energy function

$$E(x) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j)$$

where x is the label assignment for pixels. We will also use a unary potential $\theta_i = -\log P(x_i)$, where $P(x_i)$ is the label assignment probability at pixel i as computed by a DCNN. We will then divide this new image

into superpixels [1], which will be grouped by their intensity. This segmentation will not be as careful as more state of the art algorithms since intensity distributions are relatively simple groupings. We may exploit similarity of appearance and avoid needing to compare individual pixels when determining regions. On this extracted and processed image, we may then use a CNN to assign a plausible color scheme.

In particular, we identify this as a classification problem. Given an input lightness channel $\boldsymbol{X} \in \mathbb{R}^{H \times W \times 1}$, our objective will be to discover a mapping $\hat{\boldsymbol{Y}} = \mathcal{F}(\boldsymbol{x})$ to two associated color channels $\boldsymbol{Y} \in \mathbb{R}^{H \times W \times 2}$ where H, W are the images dimensions (224 x 224). We might first be attracted to the natural loss function, the L_2 regularization as in [3]:

$$L_2(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{1}{2} \sum_{h,w} ||\hat{\boldsymbol{Y}}^2 - \boldsymbol{Y}^2||$$

But this loss is not robust enough to account for the ambiguity and multimodal nature of the colorization problem. Instead, we will use the approach suggested in [7] which approaches the problem as multinomial classification. We will quantize the *ab* output space in to bins as suggested and rebalance the classes. Our new loss will be a multinomial cross entropy loss defined by:

$$L_c l(\hat{\boldsymbol{Z}}, \boldsymbol{Z}) = -\sum_{h,w} v(\boldsymbol{Z}_{h,w}) \sum_{q} \boldsymbol{Z}_{h,w,q} \log(\hat{\boldsymbol{Z}}_{h,w,q})$$

where $v(\cdot)$ is a weighting term that will be discussed in more detail in our final report. $\hat{\boldsymbol{Y}} = \mathcal{H}(\hat{\boldsymbol{Z}})$, where $\mathcal{H}$ is the *annealed mean* presented in [7]. We may be able to avoid this and map $\hat{\boldsymbol{Z}} \rightarrow \hat{\boldsymbol{Y}}$ in a simpler way, but this will be discussed in our final report.

3 Preliminary Literary References

As a preliminary reference, our group decided to reference some papers on Graph Signal Processing and their applications on processing images. We also decided to reference the Tensorflow documentation at [6].

Tensorflow is Google's open source AI framework. It is mainly used within the Python programming language, but has been adapted for Javascript as well as mobile and IoT platforms. Tensorflow is a mature platform for developing, testing and validating AI applications, with lots of detailed documentation. The Tensorflow API page that Google provides has a lot of documentation of any function we may particularly use from the library. Tensorflow has been used by many top data science companies, such as Google and NVIDIA. Tensorflow will allow us to easily make models through Tensorflow, Keras, and other python

libraries. Throughout the project, we will be referring to the Tensorflow API page for any inquiries on how to train our neural networks. We will mainly be using the tf.keras, numpy, and other frameworks to aid in our research. Tensorflow allows us to easily train a neural network given we have predefined weights and input. The plan on how we use Tensorflow for training our neural network are outlined in the latter sections of the proposal.

In our modern society, data is being used everywhere. Data is collected about your finances, spending habits, location data, and much more. Modeling these data networks/graphs is becoming increasingly complex as we add more and more extra information within these networks. As such, we need to use more complex tools to analyze these networks/graphs. One of these tools is called Graph Signal Processing (GSP) [4]. In GSP, a graph signal is "a set of values residing on a set of nodes. ... connected via (possibly weighted) edges." GSP signals are sampled by nodes of a graph, as well as represented as vectors rather than tuples or sequences. GSP has been applied to many different applications (social media, diseases, news, propagation etc.) The application of GSP we will use is Image Processing, images can be interpreted from a graph perspective, and filtering, processing and other GSP properties can be used. We can view images as a set of pixels, each associated to a vertex, which forms a regular graph where all edges have weight of 1. However, we can utilize unequal edge weights to adapt to the specific characteristics of each image.

4 Programming Environment

To complete the project, our group will primarily use two languages to analyze the dataset in addition to training the neural network. Our group will be writing our own code in a private Github repository so that we can have source control in addition to having a cloud backup just in case our code has some inconsistencies in terms of synchronization.

The first language that our group will program in would be Matlab. Matlab is a mathematical programming language that provides a lot of computational insight in a variety of applications such as data analytics, signal processing, and machine vision. For graphs, Matlab has a lot of built-in functions in order for us to analyze their properties given the weighted adjacency matrix of the graph. As discussed in the course, some properties of graphs are really useful when it comes to assigning weights to neural networks. For example, the betweenness centrality of the graph can be computed in Matlab so that we can figure out which nodes have dominance for each weight (corresponding to a color).

The other language (the primary one) that our group will use for the project would be python. Python is an interpretive programming language that is often used in machine learning applications. Some of the specific libraries, such as Keras and Tensorflow, have specialized training algorithms in order to produce

neural networks in an optimized manner. Given the image dataset (which includes black and white images with the colorized counterparts), and the corresponding color weights that were predetermined in Matlab, our group will train a neural network that would output a colorized image from a black and white input.

5 Preliminary and Expected Results

For this project the results from our Neural Network will be compared to those from a based unchanged image. At base, the most simple results that we are looking for is proper color prediction. For example, the sky and clouds stay blue/white and the roads are black. We will then create a scoring algorithm which will be compared with an unchanged color image. For example, if a pixel rgb values is (255,14,68) and the neural network returns a predicted value of (250,24,70), we can use this with a set acceptance rate to see how accurate out prediction model is. We will then continue to analyze and train the model with different weights and more image validation examples.

We predict that our Neural Network will have successfully create possible colorization. We will use MatLab to grade each image. We predict that our dataset of 25,000 images will help create a more accurate model. So far, we have analyzed an image dataset from Kaggle at [5], consisting of around 25,000 images. It includes many black and white images and their colorized versions. Our team plans to use the colorized part of the dataset as the ideal output and will produce a neural network that would take a broad range of grayscale images and colorize them.

6 Outline of Project

1. Analyze & Gather Dataset in Matlab (1 week)
   (a) Gather images of similar objects (e.g. Red Roses) to generate a specific dataset
   (b) Figure out the color weights of each of the datasets (the one on Kaggle and our own)
   (c) Generate graph with nodes of different colors
   (d) Calculate some useful dataset properties (e.g. dominant color weight, betweenness centrality)
   (e) Setup the Tensorflow Environment on a local machine

2. Implement Neural Network to train data to get desired output with Python Tensorflow (2 weeks)

   (a) Feed in the images to the neural network and be able to get some sort of image as output

   (b) Train the neural network using the image dataset from Kaggle

   (c) Train the neural network using our own set of images

   (d) Be able to feed in a black and white and get an ideal colorized output as a result

3. Tabulate Results and write Final Project Report (2 weeks)

   (a) Compute and compare the percent error for RGB values for each pixel from each neural network output with the ideal colorized image

   (b) Optimize any errors that can be lingering when training our neural networks

   (c) Document Our Findings in the Final Project Report

References

[1] Radhakrishna Achanta et al. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. DOI: 10.1109/TPAMI.2012.120.

[2] Liang-Chieh Chen et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017. arXiv: 1606.00915 [cs.CV].

[3] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. "Deep Colorization". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 415–423. DOI: 10.1109/ICCV.2015.55.

[4] Antonio Ortega et al. "Graph Signal Processing: Overview, Challenges, and Applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828. DOI: 10.1109/JPROC.2018.2820126.

[5] Shravankumar Shetty. *Image colorization*. Oct. 2018. URL: https://www.kaggle.com/shravankumar9892/image-colorization.

[6] *Tensorflow Documentation*. URL: https://www.tensorflow.org/api_docs/python/tf/all_symbols.

[7] Richard Zhang, Phillip Isola, and Alexei A. Efros. *Colorful Image Colorization*. 2016. arXiv: 1603.08511 [cs.CV].