Machine Learning based Efficient Image colorization

Manoj Nagarajan
UW-Madison
Madison, USA
mnagarajan2@wisc.edu

Arjun Gurumurthy
UW-Madison
Madison, USA
arjun@cs.wisc.edu

Ashok Marannan
UW-Madison
Madison, USA
marannan@wisc.edu

Abstract
Image colorization is a process that adds color to grayscale images. Manual image colorization is tedious and prone to human error. Existing approaches to colorize images automatically use either pixel-based scanning (expensive), are scribble-based (manual), assume strong correlation with target grayscale image, or require large number of training examples. In this project, we aim to look at approaches that perform automatic image colorization with a small number of similar reference images using Machine Learning. Initially, (1) we plan to design features which would capture different properties of a grayscale image for training ML models. Then, (2) grouping pixels into superpixels enables capturing of local information with reduced complexity. If we have a corpus of training images, (3) one or more of Machine Learning algorithms like support vector regression or k-means could be used to train an algorithm to colorize parts of images with reasonable accuracy. Based on the output of the model, (4) we perform some post-processing (like smoothing) to re-assign colors of segments predicted with low confidence. Via these four steps, we color images with reasonable accuracy using only a small subset of training images than needed in other approaches.

1 Introduction
One of major problems in the colorization process is that two objects with different colors may appear to have same color in the grayscale mode. One simple solution for this problem is to seek user inputs for colors. However, doing so will make this solution very tedious as it requires the user to repaint all the segments of the image. Another approach is to use machine learning or other similar approaches for automatic colorization, but these approaches are computationally expensive. Our approach is to efficiently automate the process by training a machine learning model on a relatively small corpus of training images. The trained model would be then used to colorize images that are similar to those in the training set. For this project, we work in the LAB color space instead of RGB, where L represents the luminance and A and B represent the chrominance. Hence the input to our algorithm is the grayscale image which is L and the output would be A and B.

2 Related Work
Levin et al. [1] proposed a method in which the user chooses colors directly and is able to refine the results by scribbling more colors in the image. A semi-automatic technique for colorization of grayscale image using a reference of colored image is described in Welsh et al. [2]. The approach uses the luminance values of neighboring pixels in the target image to fill colors from corresponding location in the reference image. In this approach the user must find a reference image containing the colors over desired regions. Larsson et al. [3] proposed a deep learning

method which is computationally expensive. In our project, we attempt to use the Levin et al technique along with machine learning approach to automatically colorize objects in the image based on some prior training data i.e. corpus of images while being faster than the deep learning approach. We also plan to comparatively analyze the performance of different machine learning algorithms.

3 Training and Testing set to be used for ML Algorithm

In order to train the machine learning model, we collected images belonging to the categories like lakes, scenic terrains and forests. We downloaded images from Instagram and Google Search Images with the appropriate tags to narrow down the image search and improve the results to avoid noise. We then scaled the pictures in our dataset to uniform sizes, and disposed of any existing grayscale photographs. We split the sample set into train and test sets in the ratio of 2:1. We verify our results with the actual colored images to estimate the performance of the algorithm.

4 Algorithm

The steps in our process can be outlined as follows:

a. Color space pre-processing

Each image in the training set is converted to grayscale and encoded in the LAB color space. L encodes luminance and (A, B) encode color information. LAB is used over RGB because the usage of a LAB color space for a machine vision implementation can give better subjective image quality than the RGB color space, similar to [5].

Also, for our training model since input is grayscale image, we can use just L component to design our feature vector and predict the (A, B) components.

b. Image Segmentation: Simple Linear Iterative Clustering (SLIC) algorithm

We found there are two approaches to color images:

1) Predicting the color of each pixel
2) Segmenting the image into superpixels and color each super pixel.

Predicting color pixel by pixel is a highly computationally intensive process. Hence we chose the second approach and use a segmentation algorithm called Simple Linear Iterative Clustering (SLIC) algorithm. Superpixels group pixels helping us to process picture highlights close to one another. They have been demonstrated progressively valuable for applications such as Image Segmentation. The segmentation approach makes utilization of Achanta et al's [4] SLIC superpixels and the DBSCAN clustering algorithm. These superpixels are then processed using the DBSCAN algorithm to shape clusters of superpixels to produce the last division.

The algorithm segments the image into superpixels using a weighting variable, relating spatial distances to shading distances, and super pixel attributes are figured out from the middle shading values. Note we utilize a relative low weighting of spatial distance to color distances in framing superpixels. Using a low weighting in SLIC algorithm decreases the possibility of under

segmenting an image and the devotion of highlight limits is better saved. The super pixel seed points have been initialized in a hexagonal framework instead of a square framework. The rationale is that this will outcome in a division that will be ostensibly 6-associated which ideally encourages any subsequent post-processing that tries to blend super pixels.

Figure 1: SLIC algorithm segments the image into multiple super pixels

c. Generation of feature vectors

For each segment, we construct a feature vector which will be the input to our machine learning classifier. We construct features from a $\delta \times \delta$ window centered around the centroid of the segment. We then perform a 2D FFT on the square to give us feature vectors. It should be noted that the square only contains the L values of the image. The shortcomings of this approach are it ignores the shape of the segment. It also considers only a small fraction of the pixels in a large segment which might not be an adequate representation. Another approach, is to construct a bounding box around the segment and select n pixels at equal intervals within the box. Pixels outside the segment but within the box are assigned L = 0. We then apply the 2-D FFT on this resultant representation.

A feature vector which is used to learn a prediction model is constructed for each superpixel segment. It consists of the following features:

- **Mean and Variance of Luminance values:** This feature has the information of the noise distribution in the segment. The mean value averages contribution of each pixel intensity in the superpixel and the variance provides information about the luminance distribution within a superpixel.
- **Centroid:** Simple properties of the image which are found via image moments include area or the total intensity, its centroid, and its orientation related information. Specifically the centroid is useful to determine segments whose location generally remains constant throughout different images. Image moments are useful to describe objects after segmentation.

To capture texture information of the segment and its corresponding neighborhood:

- **Superpixel Histogram:** Normalization of histogram is necessary as it represents probability values and is independent of the size of the superpixel.

- **Locality Histogram:** As super pixel segmentation may possibly divide the objects, the locality histogram represents information such as whether the superpixel is a part of a continuous object or whether it lies on an edge between 2 objects. The locality histogram is used get a global perspective of the current segment.
- **Gradient Magnitude:** The gradient magnitude is for capturing information about sharp changes in luminance in each $\delta \times \delta$ area around the center of the superpixel. Here we only consider the magnitude as we wish to be generic across all orientations of the superpixel.

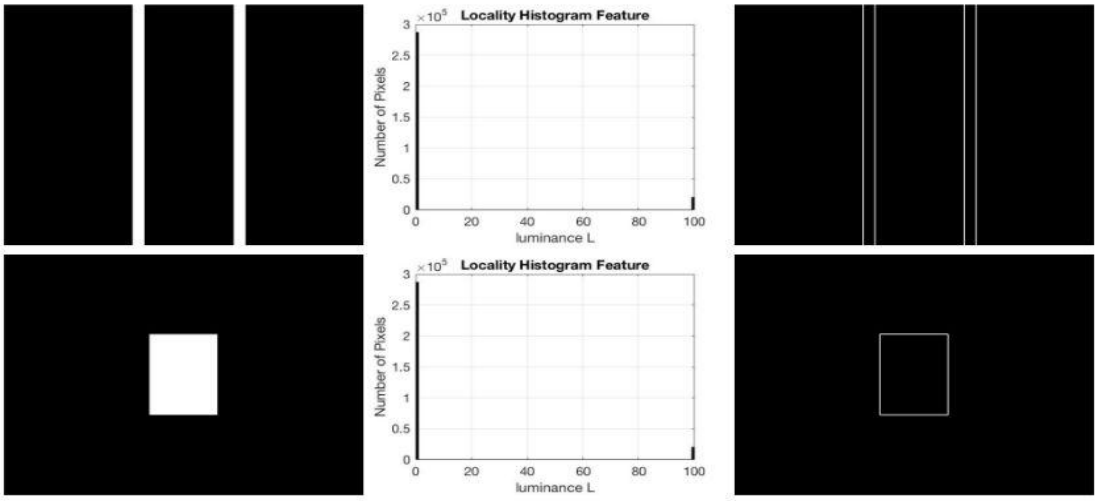


Figure 2: For each row starting from the left, the sample image, the histogram feature, and the gradient magnitude feature. The locality histogram is the same for both images, whereas the gradient magnitude is different and helps distinguish the images

d. Classifier

The code predicts 2 values for each segment - A and B and we train two models, one for A and one for B. On providing a test grayscale image, the models predict the corresponding A and B values for each superpixel. The input to both classifiers will be the feature vector representation of the corresponding superpixel. On combining them, we get the final colored image.

Which model to use?

We started off with basic linear regression for the model. Later we shifted to use the SVR model (Improved accuracy). Both SVR and Nearest Neighbour Approach is described below.

Support Vector Regression (SVR): The main idea of SVR is to fit a hyperplane such that each point is within some 'Error' of the hyperplane, while simultaneously minimizing the margin $||w||_2$. SVR centers around solving the following optimization problem

$$\min_w \quad (1 - yw^T X)_+ \qquad \text{such that} \qquad ||w||_2 < \text{Error}$$

Where X is the a matrix where each row is a feature vector and y is the corresponding chrominance value. The weight vector w defines the margin and direction of the regressed hyperplane. We attempt to minimize the hinge loss while constraining the margin to be less than a parameter 'Error'. This can be solved iteratively by using the proximal point algorithm. At each stage the update function is given by

$$w_{t+1} = \text{prox}(w_t - \tau X_T(X w_t - y)$$

Nearest Neighbour Approach: The kNN model is capable of predicting a wide variety of colors rather than SVR model. Given a set of labeled colored segments, for a test segment, we search for the best fit segment and assign its corresponding A and B values.

For computing the similarity between pixels, we compute the euclidean distances E_1, E_2, E_3, E_4, E_5 between each of the corresponding 5 features: Mean, Variance, Centroid, Superpixel Histogram, Locality Histogram and Gradient Magnitude of the test segment and a reference segment. The similarity score for each reference segment is obtained by taking a weighted combination of the Euclidean distances as shown below. The segment with the lowest score is the most similar segment and its color is assigned to the superpixel.

$$s = w_1 E_1 + w_2 E_2 + w_3 E_3 + w_4 E_4 + w_5 E_5$$

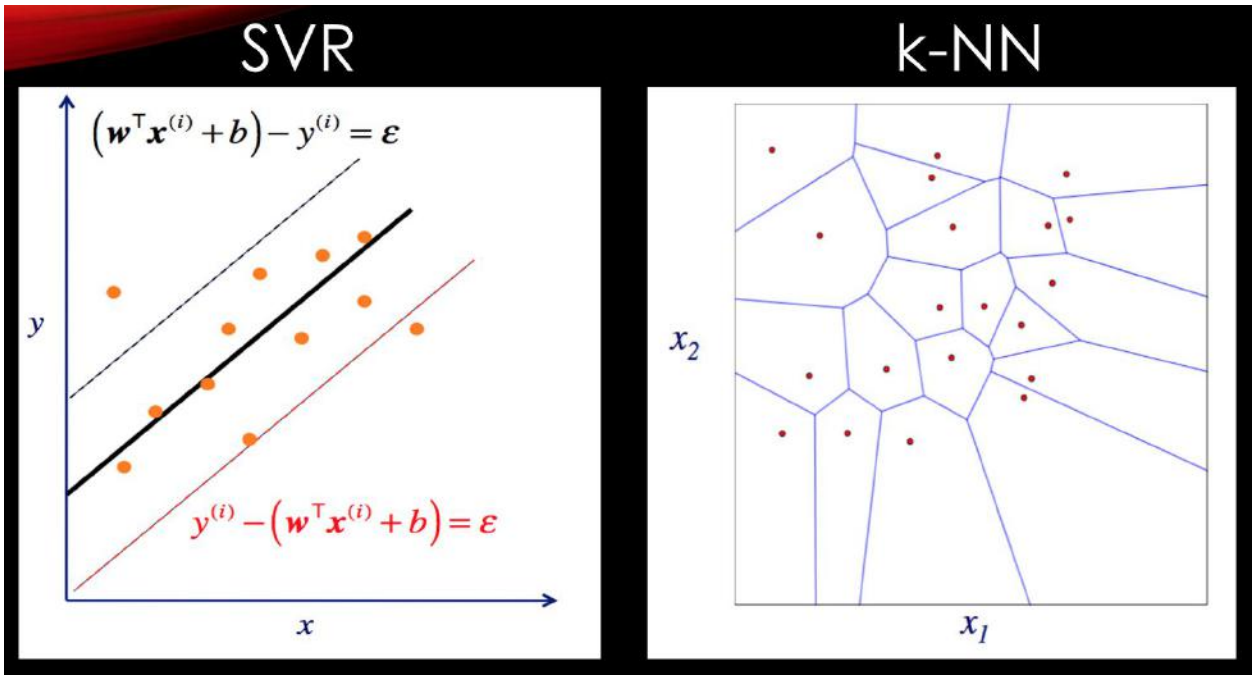


Figure 3: Support Vector Regression (left) is used to fit a line to a set of 2-d points and the boundaries of the Nearest Neighbors approach (right) in 2-d

e. Colorize

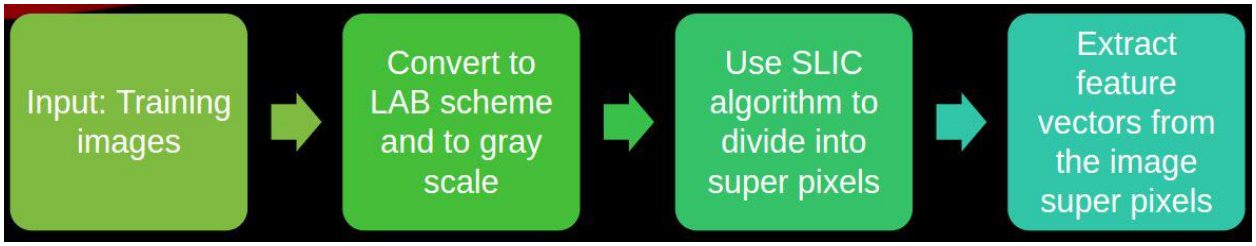
With a particular level of confidence, the model trained above outputs the chrominance values.

- ❏ **If the confidence of the prediction is too low**: the corresponding superpixel will not be colored. This ensures better accuracy and resistance to noisy labels. We then use the scribble based colorization to propagate colors to these super pixels.
- ❏ **If the confidence of the prediction is good**: The confidently colored super pixels act as scribbles. These scribbles act as a reference for propagating color to uncolored regions. The basic idea is to color similar intensity pixels with similar colors. It attempts to minimize the difference between the color of the pixel and the affinity weighted average of its neighbors.

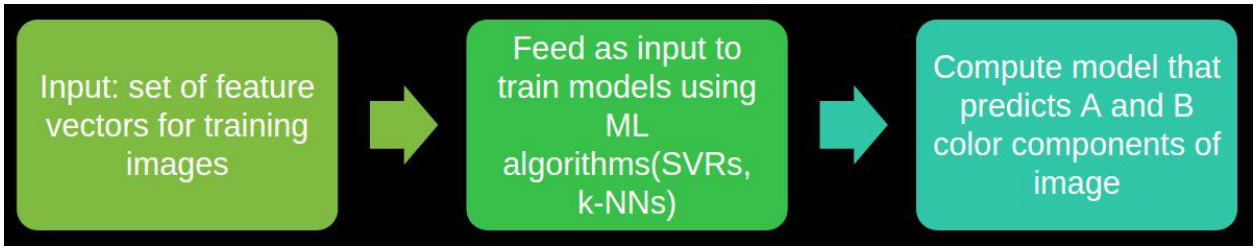
5 Flowcharts

Roughly, we have three major processes:

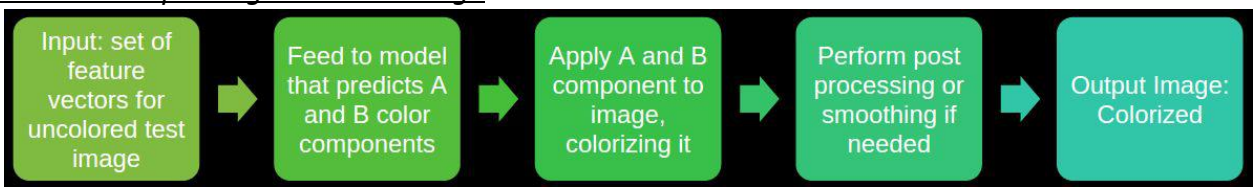
Pre-processing step for all images (for both training and predicting)



Training phase for the corpus of images



Colorization phase given a test image



6 Discussion

- ❏ **K-Nearest Neighbors versus SVR:** From the experiments, SVR failed in predicting lots of different shades of the similar colors resulting in missing out on subtle changes. On the other hand, k-NN can predict such details with high accuracy as it relies on reference images.

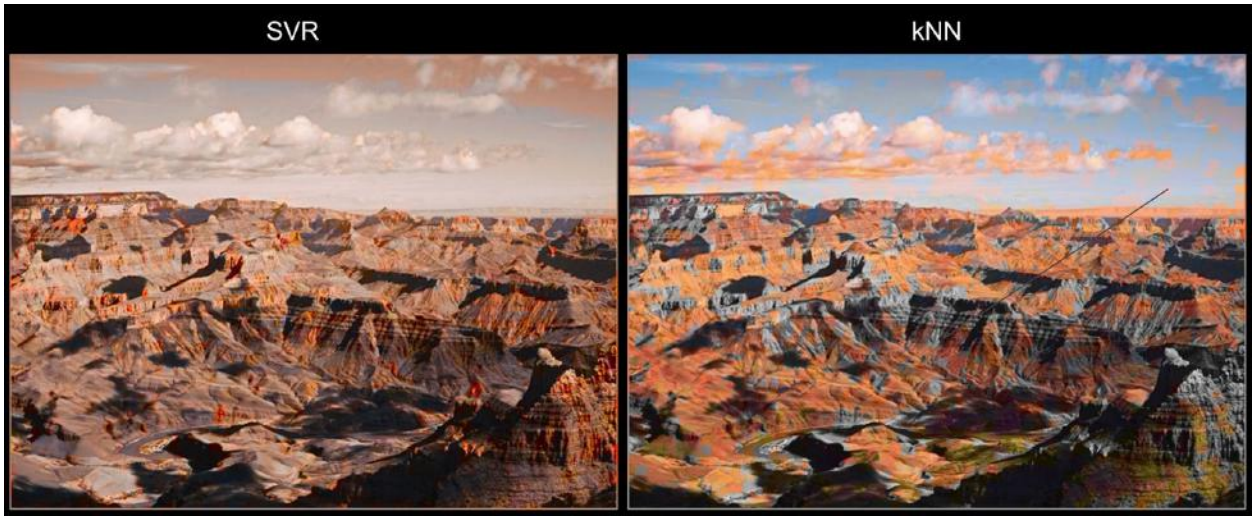


Figure 4: using SVR lacks different shades of same color. The k-NN approach recognizes more diverse colors including the blueness for the sky and greenish shades for plants.

- **So is k-NN the best solution? No, it has its limitations too:** k-NN approach is highly dependent on the reference images and hence prone to noise based on the irrelevancy of the reference images. Though It is good at predicting diversity of colors, it makes errors on very slight changes in texture of the segments and makes noisy predictions as shown in figure 6 and 9, though that has to partially account towards using very less reference images.

The below set of images shows how this algorithm works for different input images and different set of reference images.

Figure 5: The prediction is quite relevant

Figure 6: The soil is shaded with a bluish tinge as the texture difference between the end of the water and the sand is very subtle

Figure 7:The prediction is quite relevant

Figure 8 below shows a very good example that, using just a few training examples, a good colorization can be done. But the catch here is, the category (For example: Lake in figure 8) of that image should be known prior such that suitable training sets can be used, instead of using a very big generic set of training examples.

Figure 8: Using just very few training examples, this is being predicted with a very good relevancy.

Figure 9: very subtle change between the water and the shore results in the bluish shade in the image for the sand. But the difference in color between the water and the sky makes more sense as the change in texture is a bit more obvious.

Figure 10: using n=3, which is a very small training set, this relevant colorized example stands as an example that this way to colorize does a good job when there are small number of training images.

In figure 11, due to the Nearest Neighbors model's limitation with respect to subtle change in texture makes prediction errors in predicting color. Though figure 9 partly shows similar issue, the use of very less training examples there prevents it from calling the subtle chance, the only fault parameter.

Figure 11: The Nearest Neighbors model makes prediction errors in predicting color where there is subtle change in texture

7 Conclusion

In this project, we explored a new method to colorize grayscale images using various machine learning based approaches. To come up with good set of features which would accurately capture important properties of a grayscale image of each segments is a pivot. The experiments between different machine learning based models and different sets of training images helped us understand why one model performs better over the another. If used efficiently, this way to colorize can be used as an efficient domain specific colorizer where the input image to colorize is presented along with a particular domain (ex: lakes, mountains, etc) and only the images in that domain are used to train the model resulting in the usage of very less number of train set saving time and the performance.

8 Future Work

- ❏ Although this works reasonably well, there might be rough edges between adjacent pixels that could have been formed. We have to try out smoothing if that is the case. Hence if a pixel has a highly improbable color, it is flagged and re-assigned the color corresponding to the maximum likelihood estimate in that region.
- ❏ Although works reasonably well on a small set of training images, our current algorithm works better when the training set comprises of similar images. The model is susceptible to noise when non related images are introduced, which requires some internal classification.
- ❏ Evaluation and optimization of this approach in different use cases including the dynamic change in size of the train set to be used based on the model and the input should be played around with.

References

[1] Anat Levin, Dani Lischinski, and Yair Weiss. 2004. Colorization using optimization. In ACM SIGGRAPH 2004 Papers (SIGGRAPH '04), New York, NY, USA.

[2] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. 2002. Transferring color to greyscale images.In Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02), New York, NY, USA.

[3] Gustav Larsson, Michael Maire and Gregory Shakhnarovic. Learning Representations for Automatic Colorization, ECCV 2016.

[4] R.Achanta,A.Shaji,K.Smith,A.Lucchi,P.Fua, and S.Susstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods, IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI), 2012

[5] Michal Podpora,Grzegorz Paweł Korba`s, and Aleksandra Kawala-Janik. YUV vs RGB - Choosing a Color Space for Human-Machine Interaction. Position papers of the 2014 Federated Conference on Computer Science and Information Systems pp. 29–34.

Project Description

- http://pages.cs.wisc.edu/~marannan/Project_Report_CS534.pdf
- **Code**
  - Total number of lines - 2505
  - Open source libraries
    - CIRCULARSTRUCT
    - CLEANUPREGIONS
    - DBSCAN
    - DRAWREGIONBOUNDARIES
    - FINDDISCONNECTED
    - MAKEREGIONSDISTINCT
    - MASKIMAGE
    - MCLEANUPREGIONS
    - REGIONADJACENCY
    - RENUMBERREGIONS
    - SLIC
    - SPDBSCAN
    - TESTDBSCAN
- **Contribution**
  - Arjun - Machine learning methods for training models.
  - Ashok - Segmentation and feature vectors generation.
  - Manoj - Preprocess, colorize and report.