Assignment IV

Intro to Artificial Intelligence

CS 16:198:520

Kaustubh N. Jadhav
knj25

Pranav Shivkumar
ps1029

Sanyam Jain
sj770

Introduction

The basic idea behind this project is to color a grey-scale image. A color image usually consists of a matrix of 3 components, red (R), green (G) and blue (B). A pixel of the image can be represented using these components as Image[x][y] = (r, g, b), where r represents the level of red, g represents the level of green and b represents the level of blue in the pixel of the image and each has values ranging from 0 to 255. We have implemented this project using Python.

The image that we have chosen to perform this is shown below:

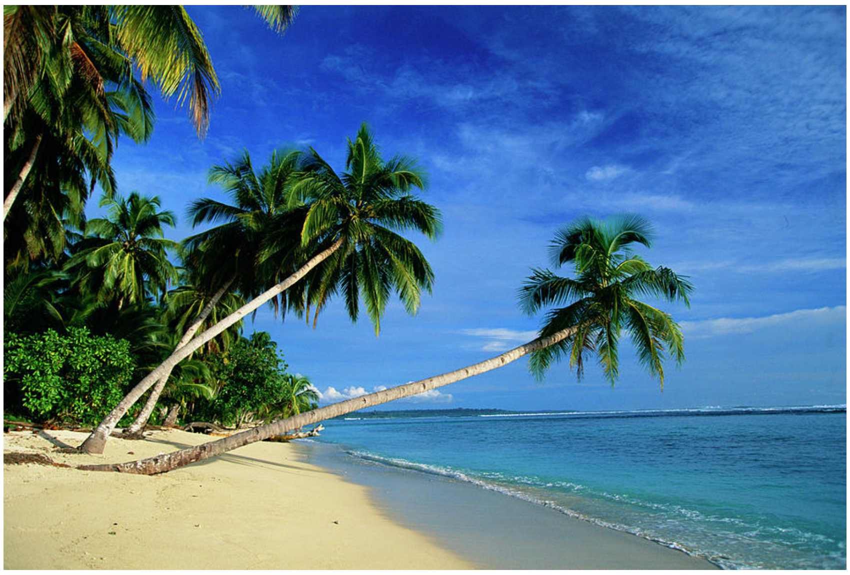


Figure 1: Original Image

We resize this image in order to reduce the resolution of the original image, the resized image is shown below:

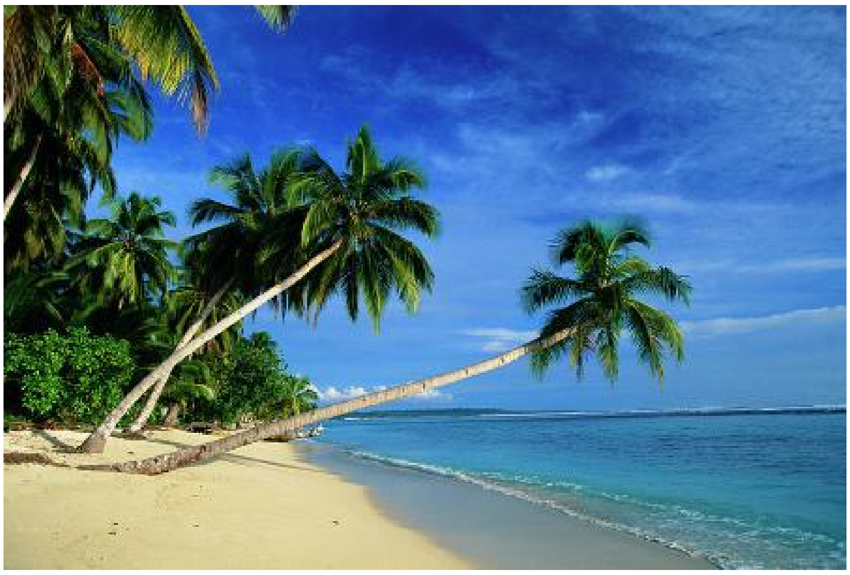
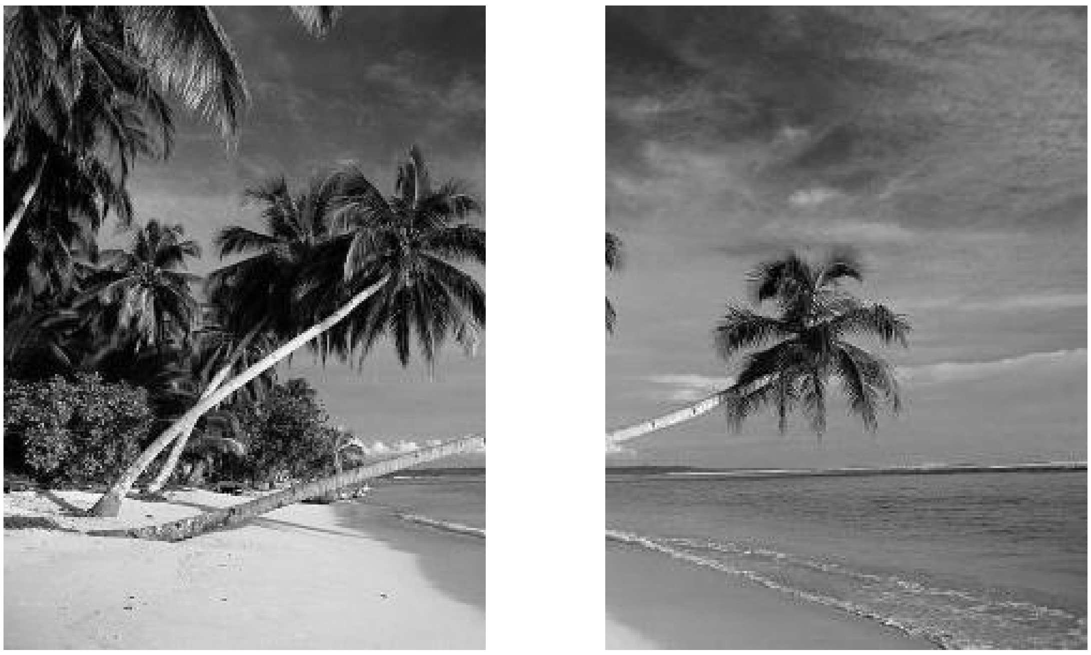


Figure 2: Resized Image

Now in order to convert this to grey-scale, we use the classical conversion formula, given by:

$$Gray(r, g, b) = 0.21r + 0.72g + 0.07b \tag{1}$$

Effectively, this converts the image from a three dimensional matrix to a single vector of values ranging from 0 to 255, with each value corresponding to a particular shade in the grey-scale image. Now, our aim is to color this grey-scale image. In order to do so, we use the left half of the image as the training data and the right half of the image as the testing data. The training input image and the test input images are shown below:



(a) Training Image (b) Test Image

Figure 3: Gray Scale Images

Part 1

We use 2 agents in order to color the image:

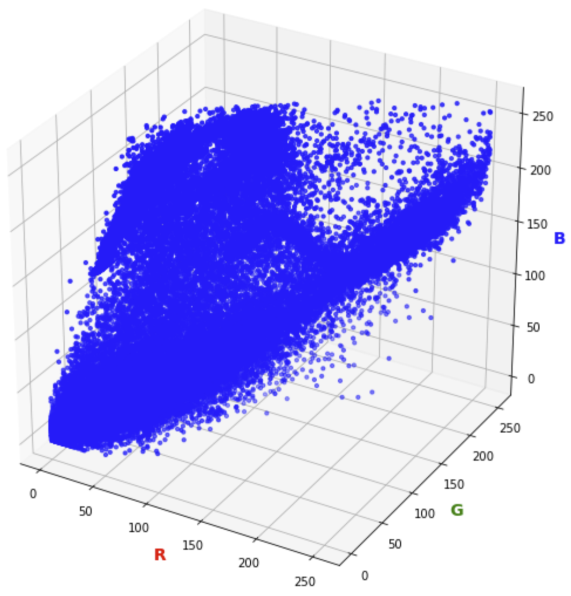
- Basic Agent
- Improved Agent

Implementation

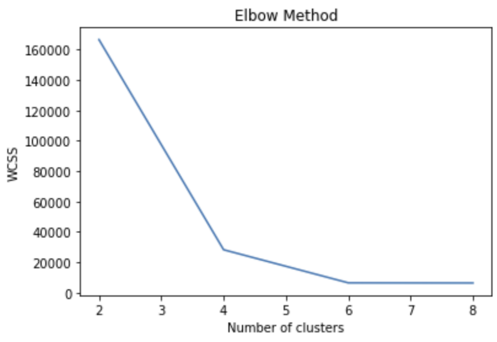
Reducing Color Space

To make coloring easier, we reduce our color-space. To reduce color-space we are using the k mean clustering algorithm on colors (R,G,B) in the Image. The given value of k is "5", but to verify that, we used the elbow method. In Figure 4(a), each point represents a color in our original image; thus we use the K mean clustering algorithm to bring down these number of colors. In Figure 4(b), we plot the Within-Cluster-Sum-of-Squares (WCSS) vs number of clusters for the colors data. As shown in the figure, WCSS stops improving beyond k = 6 where there is an "elbow", so we can conclude that k = 6 is the best value of k for k mean clustering algorithm.

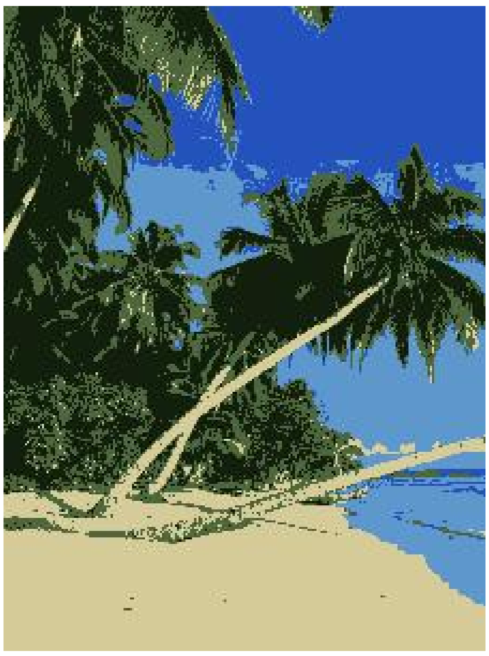
We decided to go with k = 5 as it took less time and WCSS is fairly close to k = 6. 5 cluster centers are used to recolor the images and we will we using these re-colored images as our expected output for training and testing data.



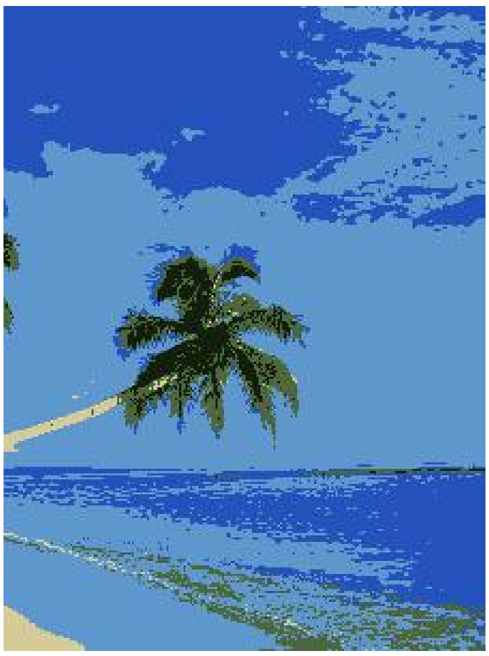
(a) Original Color space



(b) WCSS vs Number of clusters



(c) Re-colored Output Training Image



(d) Re-colored Output Testing Image

Figure 4

Basic Agent

The basic agent is an instance-learning based agent. The logic behind this agent is to compare the grey pixel patch (3X3) around the concerned pixel in the input test data with a set of patches in the input training data, find the best six patches and use the majority color from the training output data of these patches to color the concerned pixel in the input test data. Below are the information regarding the basic agent.

- **A specification of your solution, including describing your input space, output space, model space, error or loss function, and learning algorithm.**
  For this agent, the input data is the 3X3 clusters for each pixel in the gray scale image and the output data is the color (from 5 colors) of the center pixel in the cluster. On training input data, we performed k-mean clustering on grey-scale input images with k = 50. So, each patch in original input data is replaced with one of the 50 centers.
  Model: For each value (3x3 pixel patch) in the test data, we first identified the cluster corresponding to the cluster created for the training data, then, in that cluster, we identified 6 closest patches to the test data value using Euclidean distance. For these 6 selected patches, we get output value from training data and majority color is selected, if there is no majority color, then the color output for the patch closest to test input data is selected.
  Error Function: Mean Squared Error is calculated for each output color image and expected color image. Also, accuracy, which is measure of how many pixels are identified correctly, is calculated. For this basic agent, accuracy is low as 20%

- **How did you choose the parameters (structure, weights, any decisions that needed to be made) for your model?**
  For the basic agent, most of parameters were given. Number of patches to compare were taken as 6, the value of k for k-mean Clustering for input gray patches is taken as 50.

- **Any pre-processing of the input data or output data that you used.**
  The only pre-processing that was necessary was creating cluster of training input data.

- **How good is the final result? How could you measure the quality of the final result? Is it numerically satisfying, if not visually?**
  The final result was not visually satisfying as this basic agent just picks the best color based on "similarity" to training data. This agent does not identify any of the features on the basis of which coloration is done. So visually, it is really hard to see any form in the output image. Numerically also, this agent was only able to identify only 20% of pixels correctly.

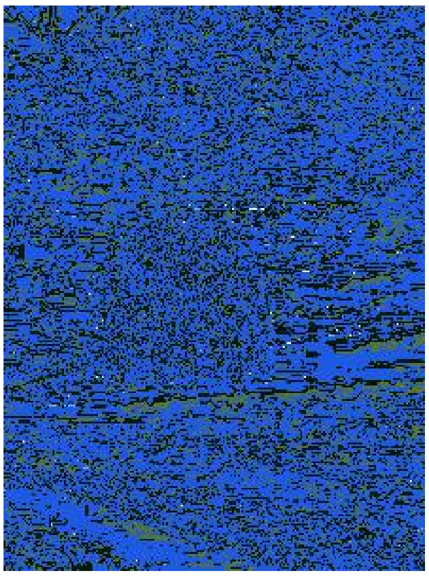


Figure 5: Output Image for basic agent

Improved Agent

For the Improved agent, we implemented the softmax regression. Based on the input data, the agent decides the pixel color out of 5 choices. Below are the information regarding Improved agent

- **A specification of your solution, including describing your input space, output space, model space, error or loss function, and learning algorithm.For instance, do you want to attempt this as a classification problem (as in the basic agent) or a regression problem, or some kind of soft classification?**

Like the basic agent, the input space is a 3x3 grey scale pixel patch for each pixel in the input data and the output space is the color of the concerned (center)pixel.
Model : The model used is the softmax function, defined as:

$$\phi_{\text{softmax}}(z^i) = \frac{e^{z^i}}{\sum_{j=0}^{k} e^{z_k^i}} \tag{2}$$

Loss Function : We used the cross entropy loss function with the regularization term as loss function. Below is the equation for that.

$$Loss = \frac{-1}{N}\sum_{i=1}^{m}[Y * log(softmax(X.W))] + \frac{\lambda}{2}\sum_{i=1}^{m} W.W^T \tag{3}$$

In the above equations, Y, X, W are the matrices that are being used in the computation.

- **How did you choose the parameters (structure, weights, any decisions that needed to be made) for your model?**
We have to decide hyper parameters like learning rate(α), regularisation parameter(λ) and batch size for Stochastic Gradient Descent. To decide the values for these parameters, we did grid search and the parameters that gave the best result on the test data were selected.

- **Any pre-processing of the input data or output data that you used.** Input data was standardized for this implementation, a columns of ones was added to include bias as in feature parameters.

- **How did you handle training your model? How did you avoid over-fitting?** We used Stochastic Gradient Descent for training. To avoid over fitting, we followed three measures: Stochastic Gradient Descent (with mini batch), Termination based validation loss (we had validation data apart from training data) and L2 norm Regularization.

- **How good is the final result? How could you measure the quality of the final result? Is it numerically satisfying, if not visually?**
The final result for the improved agent was better than the basic agent both numerically and visually. This agent was able to identify "objects" in the image by classifying them as on basis difference between grey scale values. It was able to classify "light" and "dark" shades in image. But, the issue with this agent is that even though it is multi-class classification, the agent essentially colors dark shades with one color and light shades with one color. The accuracy for this agent is 80%

- **An evaluation of the quality of your model compared to the basic agent. How can you quantify and qualify the differences between their performance? How can you make sure that the comparison is 'fair' ?** Both agents were evaluated on the basis of number of pixels with expected color values. Since both agents were identifying the same output, we can say that this comparison is fair.

- **How might you improve your model with sufficient time, energy, and resources?** One obvious thing by which this model could be improved is that, instead of just using softmax regression, we could use neural networks. This will help us to identify different levels of lightness (or darkness) in the image and do coloration according to that. Given the time and computing power to train a highly layered neural network we think we can improve coloration. But one thing that we have to keep in mind is that same grey-scale level can have different color in different images so to develop a good generalise model we will require a large and diverse data set and again time and resource to train on large data set.

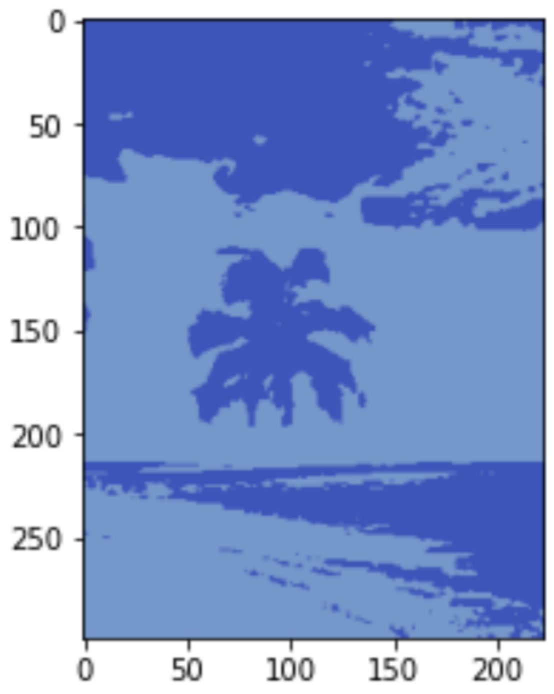


Figure 6: Output Image for Improved agent

Part 2

Implementation

Expanding on our work in Part 1, we decided to create a coloring agent using Convolution Neural Networks (CNNs). For this we primarily use Keras library to create, train and predict from the model. We did three implementation

- Predicting (R,G,B) values from grey scale values using logistic regression on data set of 500 images
- Predicting (a,b) values from L channel values (equivalent to grey scale values) using logistic regression on data set of 500 images[1].
- Predicting (a,b) values from L channel values (equivalent to grey scale values) using logistic regression for 1 image used in Part 1. We decided to for one image as for multiple images even though our accuracy was about 60% on test data with multiple images coloration result we not satisfying. Images were either colored with shade red or green so we decided to train a model for single image only. For this model we used left half of image as training data and right half as testing data.Below is the output for single image.

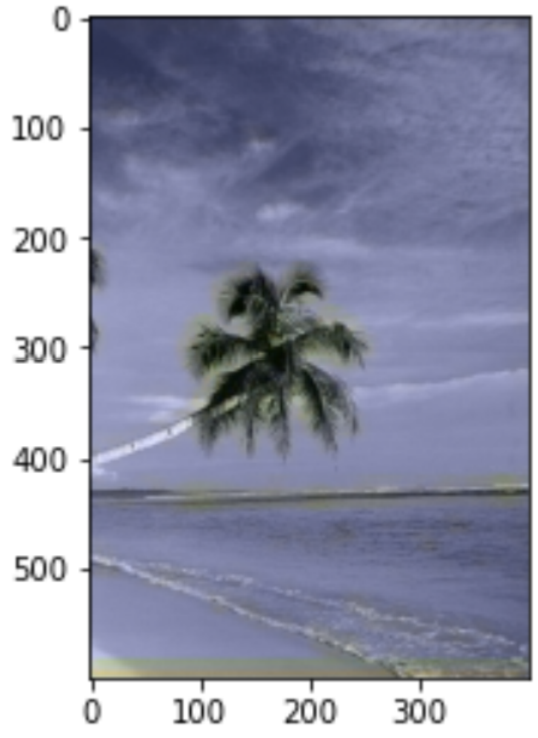


Figure 7: Coloration output based CNN coloration

Below are the information regarding the CNN based agents

- **A specification of your solution, including describing your input space, output space, model space, error or loss function, and learning algorithm.For instance, do you want to attempt this as a classification problem (as in the basic agent) or a regression problem, or some kind of soft classification?**

  Input space for first model is grey-scale value of each pixel and output are R,G,B values. While of other two models Input space is L channel value and output space is a, b channel values. For these agents we combine input and output to get RGB image from LAB image.

  Model : All CNN based model used logistic regression only difference is While first model predicts (R,G,B) values based from grey scale values other two models predict (a,b) channel values based L channel value. For all agents Loss function is Mean Square Error.

- **How did you choose the parameters (structure, weights, any decisions that needed to be made) for your model?**
  For all models we stick to basic hyper parameters such as optimizer was set to "adam". But first two models we ran these models to with multiple batch sizes to check accuracy and stick one one with reasonable accuracy in decent amount of time.

- **Any pre-processing of the input data or output data that you used.** For fist agent all values i.e input grey scale values and output RGB values were scale b/w 0 to 1 by dividing original values by 255.
  For second agent input values were scaled b/w 0 to 1 and output values were scaled b/w -1 to 1 values (not shown in code and we converted our RGB images to LAB values and stored them prior to our use in Jupyter notebook)
  For last agent output values were scaled b/w -1 to 1.

- **How did you handle training your model? How did you avoid over-fitting?** To prevent over fitting we use early stopping with patience level(no. of epochs), thus if no improvement in loss observed with patience level training will terminate.

- **How good is the final result? How could you measure the quality of the final result? Is it numerically satisfying, if not visually?**
  The final result for the CNN based agents 1 and 2 were not satisfactory as visually though numerically they showed good accuracy(60%). But CNN based agents 3 showed result both visually and numerically better than all previous agents as image maintained its integrity and coloration was also satisfactory. The accuracy for this agent on test data was 86.%

- **An evaluation of the quality of your model compared to the basic agent. How can you quantify and qualify the differences between their performance? How can you make sure that the comparison is 'fair' ?** Comparing these models with models in part 1 is bit unfair as these models are trying to predict all colors values directly grey scale values, thus these models are more generalise than models in part one.

- **How might you improve your model with sufficient time, energy, and resources?** We will be more interested in improving agent 1 and agent 2 by experimenting with model structure, add more layers and a better dataset. So given sufficient time and computing power we will like to train these models in better way.

Code Overview

We have implemented each section of these projects using Jupyter notebooks. The notebook details are listed below:

1. *Project 4 Image Recoloration - Part 1a* - This notebook implements the k-means clustering algorithm and the basic agent for re- colorization.

2. *Project 4 Image Recoloration - Part 1b* - This notebook implements the improved agent for re-coloration with softmax regression.

3. *Project 4 Image Recoloration - Part 2 Multiple Images 1* - These notebooks use the Keras library to implement the Convolutional Neural Network (CNN) to recolorize multiple images by using the LAB values.

4. *Project 4 Image Recoloration - Part 2 Multiple Images 2* - This notebook recolorizes a single image, using the convolutional neural network from the Keras library by direct conversion of the gray-scale image to RBG.

5. *Project 4 Image Recoloration - Part 2 Single Image* - This notebook recolorizes a single image, using the convolutional neural network from the Keras library.

Responsibilities

GitHub Repository : Intro to AI - Project 4

For this project, all the team members shared equal responsibilities, but each one of us took lead in the below mentioned parts of the project:

Kaustubh N Jadhav (knj25) - Readability of Code
Pranav Shivkumar (ps1029) - Report and Readability of Code
Sanyam Jain (sj770) - Coding, Algorithms and Answers for the report.

References

Colorful Image Colorization - Richard Zhang, Philip Isola, Alexei A. Efros