CS 440 **Assignment 4 - Colorization**
Spring 2021 **GITHUB:** https://github.com/Norden-Tenzin/440ArtificialIntelligence

---

Name: (Sangkyun Kim) **NetID**: (sk1998) **Section**: (05)
Name: (Tenzin Norden) **NetID**: (tn266) **Section**: (08)

---

1. **Instructions**

   (a) To RUN:

   i. Use arguments:

   A. For argv[1]:
   "-cmd": This command requires other commands to be used with it. it uses the terminal to output its data.

   A. "basic": This runs the Basic algorithm. After completion Displays the image and saves it.

   B. "advanced": This runs the Advanced algorithm. After completion Displays the image and saves it.

   Examples:
   "python main.py -cmd basic"
   "python main.py -cmd advanced"

2. **Basic Agent**

   (a) **Specification**
   We select 5 random position on the left image file to initialize out centroids. The next step is grouping all the pixels based on distance between the target pixel and the each centroid. To explain, Whenever a pixel found its nearest centroid, it was classified as belonging to that centroid. To calculate the distance, we use euclidean distance(np.linalg.norm(begin - end)). Then, for each cluster we calculate the average value(new centroid) and compare with old centroid. We repeatedly run this process until there is less than 5 difference between old and new controid (less than 5 difference of each R, G, B values between old and new centroid).

   When there is no big difference, the program stop and recolor the left side image with this 5 color from centroid. Then the program create 3 x 3 patch from this recolored image. To recolor right side grey scale image, each patch(3 x 3) from right side is compared with patch from left side and find most nearest 6 patch from left.

   (b) **Result**

Since we only use 5 centroid(5 color), the result of basic agent too different from original image. Even compare with recolored left side, the right side image is rough.

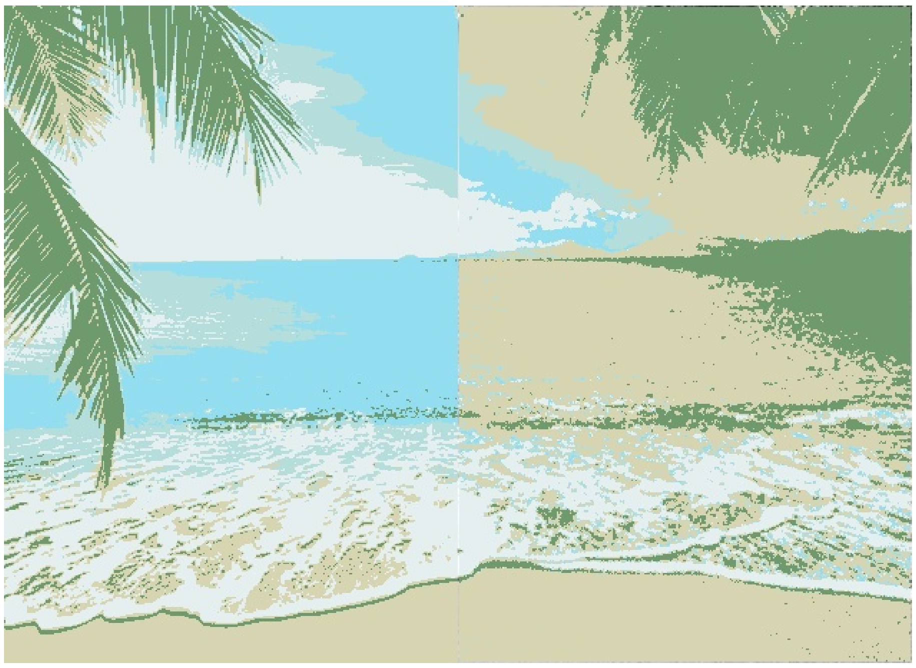


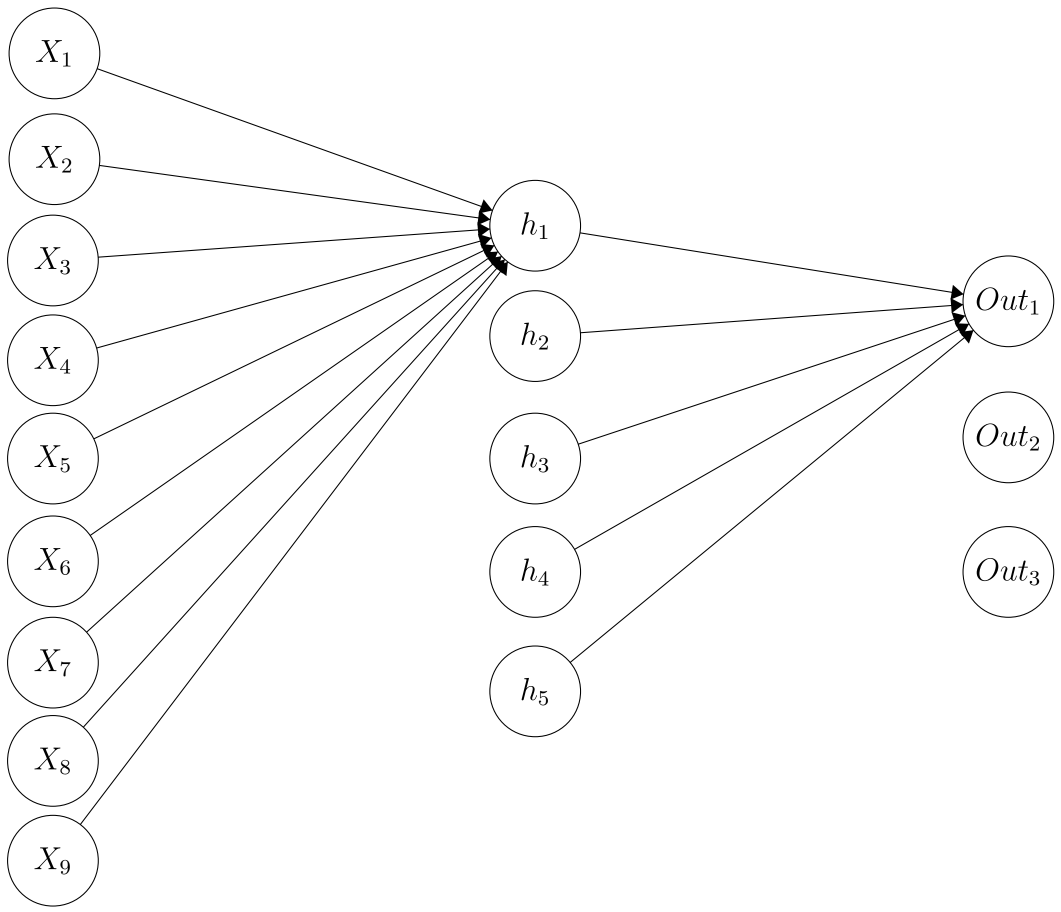
Figure 1: Result of basic agent

3. **Advanced agent**

   (a) **Specification**

   For the advanced agent, we select neural network and the basic idea is from the lecture note "An Example Network". The left side image is training data and the right side image is test data. Our neural network has 3 layers (input, hidden, output). Input layer has 9 nodes because each patch has 9 pixel data. The hidden layer has 5 nodes. We select 5 nodes in the hidden layer as mentioned in the lecture note. The output layer has 3 nodes since pixel has R, G, B three data corresponding to the middle pixel in the 3x3 patch.Given 9 input nodes, we need 45 weights to calculate the 5 nodes in the hidden layer. Then we need 15 nodes for calculating 3 nodes in the output layer. To calculate each layers nodes, we use sigmoid function as activation function.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



For using gradient decent, we calculate each partial derivative of the loss function by using equations from the lecture note.

$$\frac{\partial L}{\partial Out_C^2} = 2(\partial Out_C^2 - yC) \tag{1}$$

$$\frac{\partial L}{\partial Out_k^1} = \sum_{C=0}^{2} \frac{\partial L}{\partial Out_C^2} \sigma'(w^c Out^1) w^{C_k} \tag{2}$$

$$\frac{\partial L}{\partial Out_k^C} = \frac{\partial L}{\partial Out_C^2} \sigma'(w^c Out^1) Out_k^1 \tag{3}$$

$$\frac{\partial L}{\partial Out_i^k} = \frac{\partial L}{\partial Out_k^1} \sigma'(w^k Out^0) Out_i^0 \tag{4}$$

The equation 3 describes the weight derivatives between the hidden and out layer. The equation 4 describes the weight derivatives between input and hidden layer. The first term of the equation 3 and 4 are from equation 1 and 2. We randomly generates and initialize 60 weights(45 + 15) as values 0 1. The information regarding each patch include the 9 gray scale values with the coordinates of the middle pixel.

We also divide all the patches into a specific number of groups to train our model on a group several times before train the model with next group. This process perform better and make our result more accurate. To train the model, we calculate nodes based on initial weights. We then used gradient descent to find changes between old weight and new weight. Keeping this data, we then continuously run this process for all the remaining patches. We then adjusted our weight based on the average of derivatives, and repeated this process using the same data, but the new weights, for 100 times. We then repeat this whole process for the rest of data. When the model was trained, we use this trained weight values to recolor right side grey scale image.

The model make overfitting when there is not enough data because when there is lack of data, the model can remember some specific pattern or noise easily. Hence, we can avoid overfitting by increasing the quantity of data(Data Augmentation). We divide data into several group and run the process through this group continuously and this increase the quality of performance.

(b) **Result and comparison**

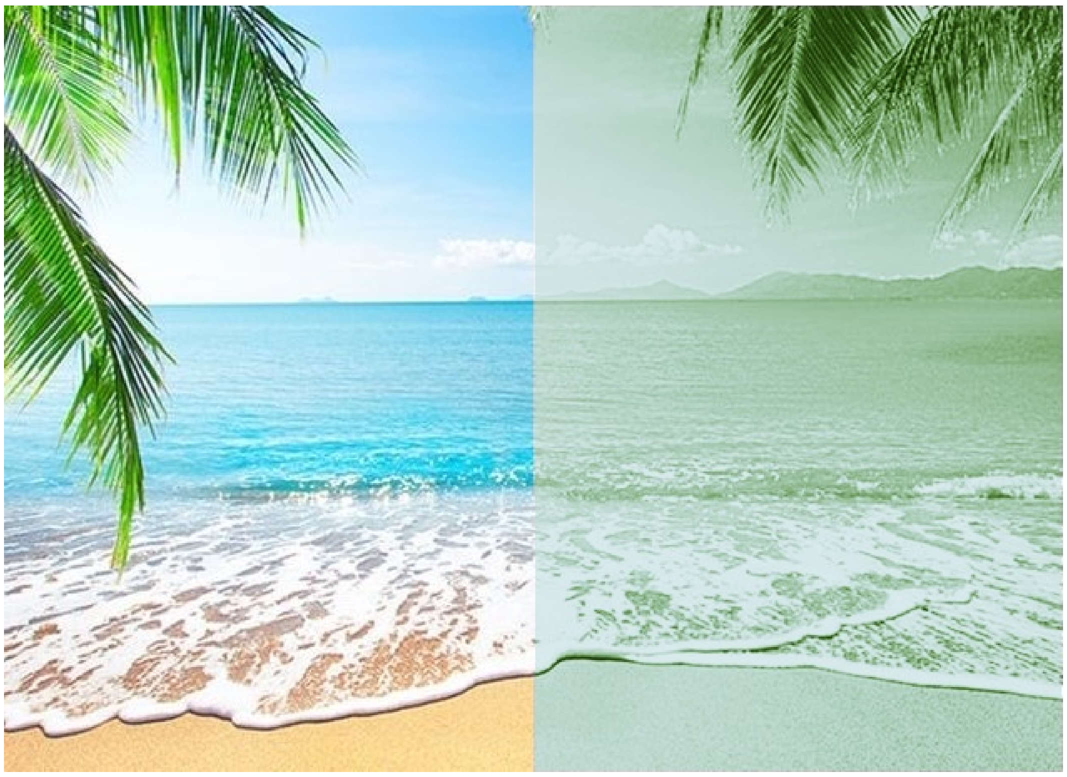


Figure 2: Advanced agent

We can see that the neural network makes much smoother and realistic image. All the objects in the result of neual network has clear shape. However, result from the basic agent has grainy shape and colors. Even the neural network does not match the real exact color, it does better job at creating a realistic image.The basic and the advanced agent create different results because each agent optimize their result based on the different parameters.

(c) **Improvements**
We can improve the model by train the model with several different images. Allowing a broad range of data helps the model to calculate more accurate weights. In addition, using another hidden layer may be useful because this allows the network to pick out some of he finer details missing from our result.

4. **Acronym**

(a) C.A.N.N
Colorizing Advanced Neural Network

5. **Summary**

(a) We split the work 50 50 and worked on the entire project together. We had daily meetings to discuss our progress. We used github as a collaboration tool. We would talk about the issue at hand before tackling it and try to break down the problem into smaller more easy to solve problems.