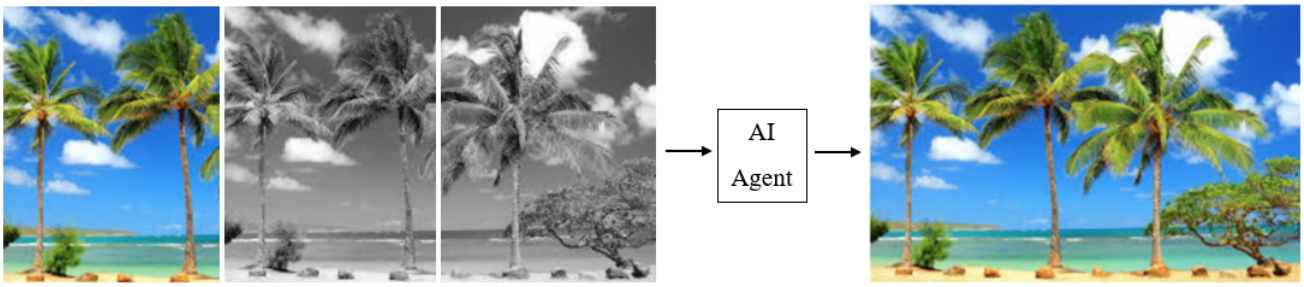
RUTGERS UNIVERSITY

Colorization of Grayscale Image

Prathamesh Kulkarni
December 15, 2020

1 INTRODUCTION

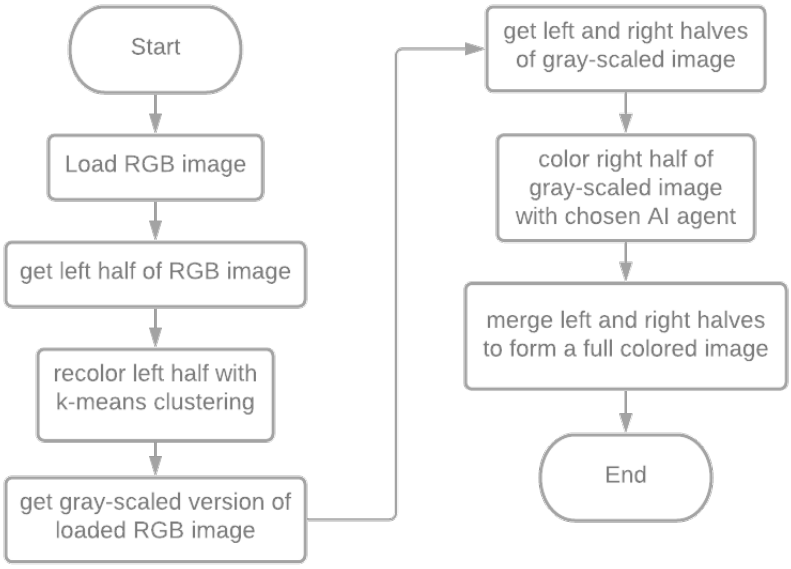
A colored image is a 3-dimensional matrix of numbers where the shape of a matrix is width x height x 3. Each number in this matrix is an integer varying between 0 and 255. This image is converted into a gray-scale image with potential loss of information about the colors. Task of an AI agent is to recolor the gray-scale image as accurately as possible. For any colored image, colored as well as gray-scaled left half and gray-scaled right half of the image are given as input to the AI agent. Agent does not have information about the true colors of the right half of an image. It has to use information about the true colors of left half of an image and color the gray-scaled right half as accurately as possible. There are two AI agents - Basic Coloring Agent and Improved Coloring Agent that color the gray-scaled right half. These agents differ in the strategies/ algorithms used to do the colorization.



Colored Left Half Gray-scaled Left Half Gray-scaled Right Half | Colored Full Image

2 System Design

2.1 System Architecture



2.2 Important Modules

1. **Image Preparation**
   An RGB image is stored on a machine locally. This module begins by loading this colored image from a given path into a 3-dimensional ndarray. As AI agent gets only the left-half of the colored image, it uses slicing on ndarray to fetch the left half of it. As AI agent also gets the left and right halves of gray-scaled image, it converts colored image into gray-scaled image and uses slicing to fetch the left and right halves. Image preparation ends with 3 halves - colored left half, gray-scaled left half, gray-scaled right half.

2. **Recoloration of Left Half of an Image**
   Colored left half can have a large number of colors. In order to narrow down the range of colors, it executes k-means clustering algorithm on the colors present in the left half to determine the best k representative colors. It recolors every pixel in the colored left half with the centroid color that it is clustered with.

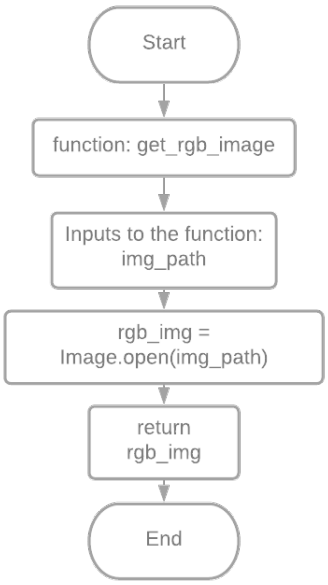
3. **Coloration of Right Half of an Image by Basic Agent**
   Basic agent treats coloration of right half of an image as the hard classification task. It uses K-nearest neighbor algorithm to color the gray-scaled right half of an image. For every 3 x 3 gray-scaled patch (test patch) in the right half, it finds 6 nearest neighbors (most similar 3 x 3 patches) in the gray-scaled left half (training data). If there is a majority representative color amongst the colors of middle pixels of corresponding patches in recolored left half, then middle pixel of the test patch is colored with this representative color. If there is no majority representative color, then middle pixel of the test patch is colored with the color of a middle pixel of the most similar patch from recolored left half.

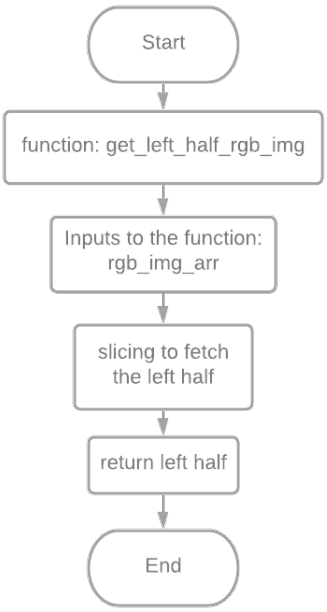
4. **Coloration of Right Half of an Image by Improved Agent**
   Improved agent treats coloration of right half of an image as the soft classification task. Instead of K-nearest neighbors, it uses feedforward neural network that is trained using backpropagation algorithm to color the gray-scaled right half of an image. It trains neural network on the dataset created from gray-scaled left half and corresponding recolored left half. When a flattened test patch is given as input to the neural network, argmax of the vector of values in the output layer determines the centroid color number with which to color the middle pixel of a test patch.

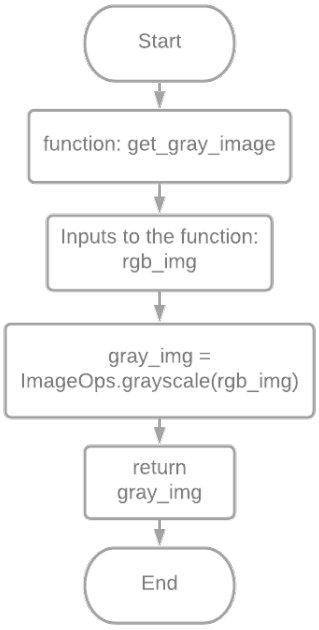
3 Detailed Design

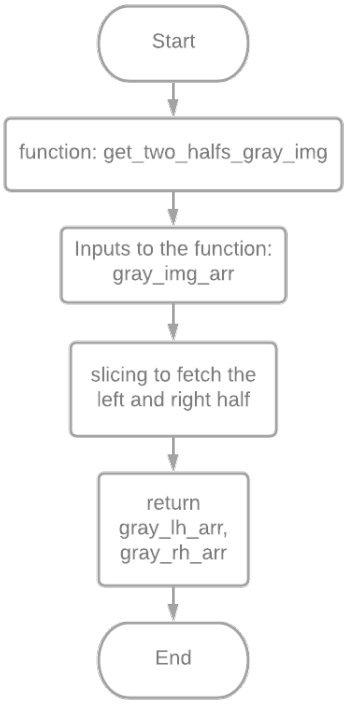
3.1 Image Preparation

It begins by loading an RGB image that is stored locally. It loads colored image from a given path by using Image class of Python Image Library (PIL). It converts this colored image into 3-dimensional ndarray object and uses slicing to fetch the left half of it. It also converts the colored image into gray-scaled image by using grayscale function in ImageOps class from PIL. This function uses g = 0.299 * r + 0.587 * g + 0.114 * b to convert each RGB pixel into gray-scaled pixel. Left and right halves of gray-scaled image are fetched by performing slicing operation on ndarray object. Image preparation ends with following 3 images - colored left half, gray-scaled left half and gray-scaled right half. Gray-scaled left half and colored left half together form the training set of AI agent. Gray-scaled right half forms the test set of AI agent.



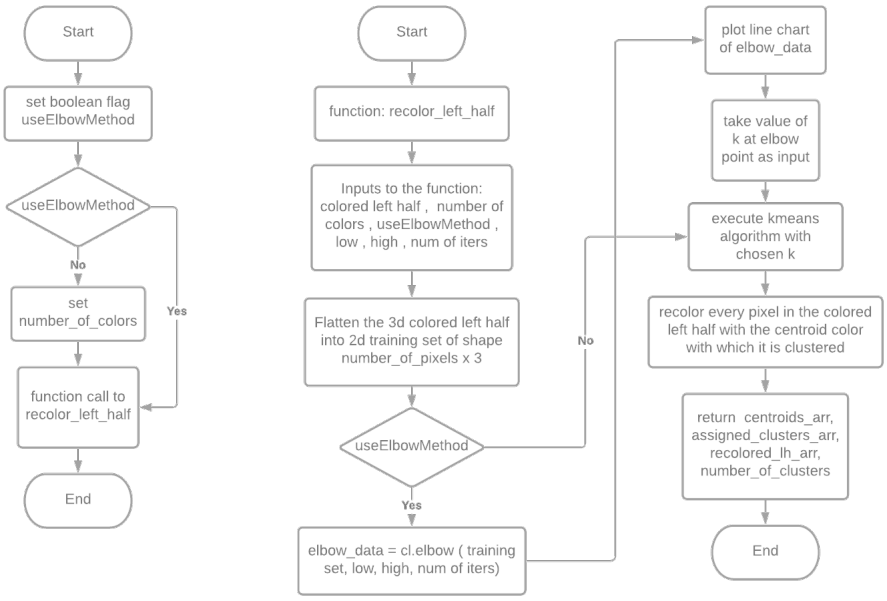




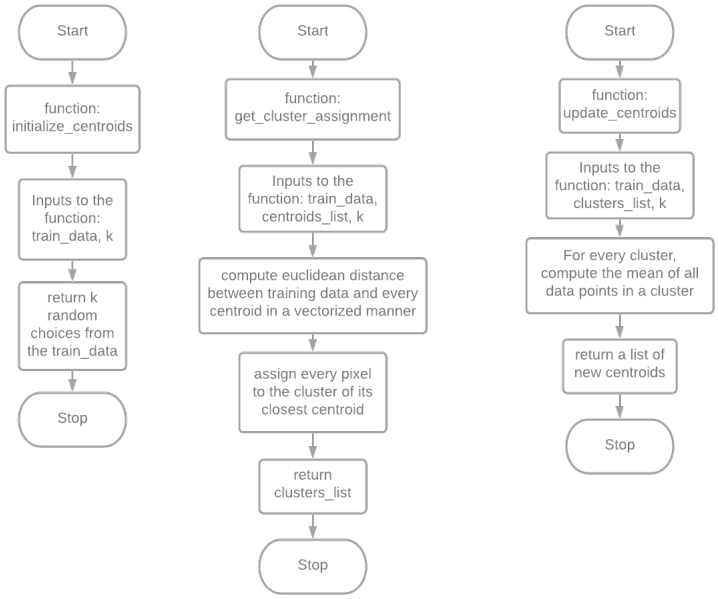


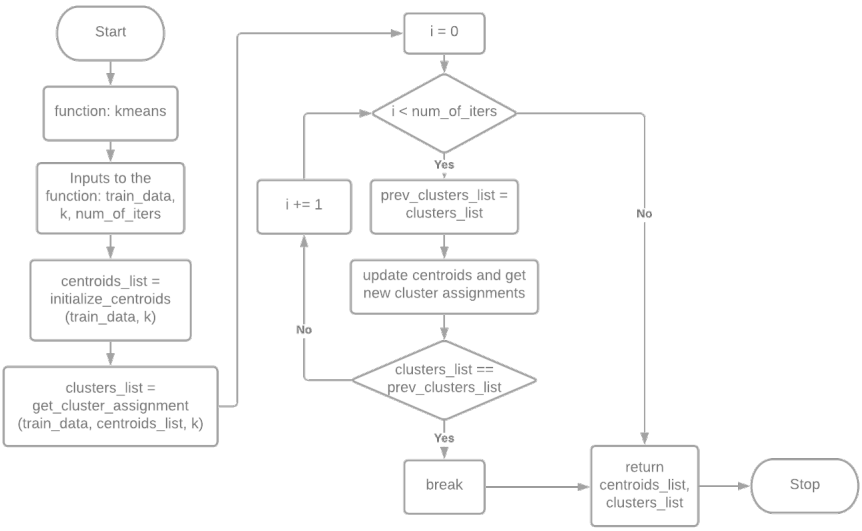
3.2 RECOLORATION OF LEFT HALF OF AN IMAGE

As the number of colors (number of classes) in the colored left half of an image can be much larger, it is necessary to narrow down the range of colors so that AI agents can treat coloration of gray-scaled right half as the classification task. Recoloration of left half of an image is done by using k-means clustering algorithm. It takes colored left half as training data and executes k-means clustering algorithm to determine the k centroid colors that are later used to recolor the left half. It has the flexibility to use elbow method for a given range of values of k and take value of k at elbow point in the plotted line chart as input. Once the centroids and clusters are computed by kmeans clustering algorithm, every pixel in the colored left half is recolored with the centroid color with which it is clustered.

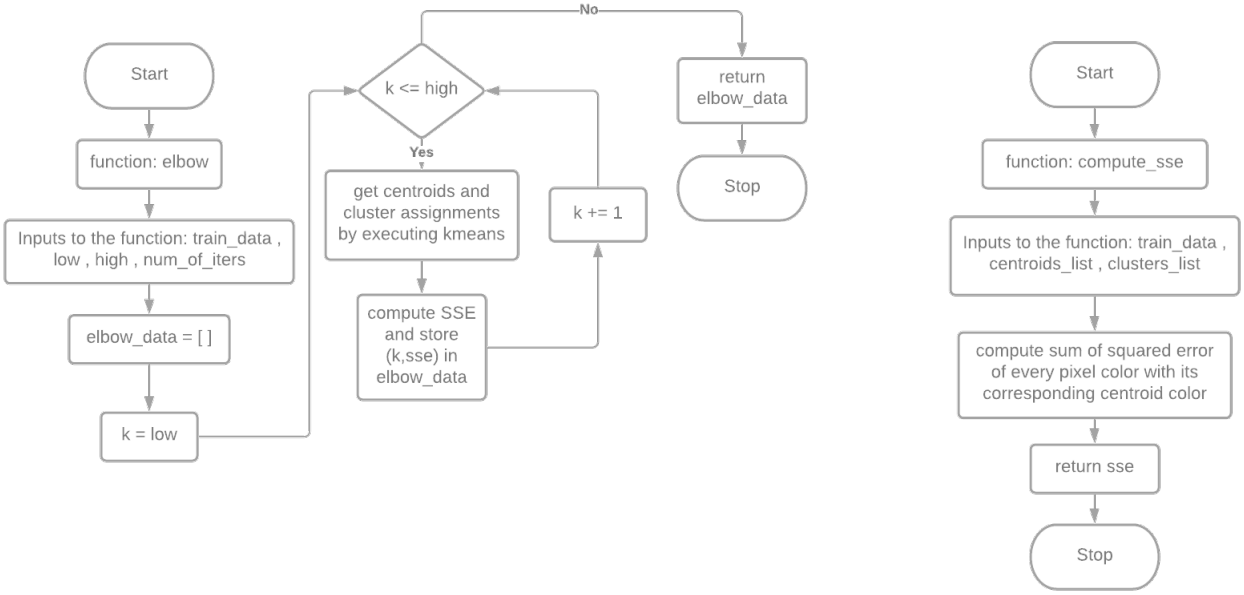


k-means clustering algorithm begins with the initialization of centroids to randomly chosen k examples in the train data. It then iteratively performs two operations - get cluster assignments and update centroids. To compute cluster assignments, euclidean distance of every pixel color in the training data with every centroid color is computed. To speedup this computation, vectorization capability of numpy is used instead of explicit for loops. Every pixel gets assigned to the cluster of its closest centroid. Once every pixel is assigned to a particular cluster, centroids need to be updated. For every cluster, centroid of a cluster gets updated to the mean of all the data points (pixel colors) assigned to that cluster. Iterations are performed until either of the following events occur - iteration counter reaches given number of iterations or there is no change in cluster assignments.

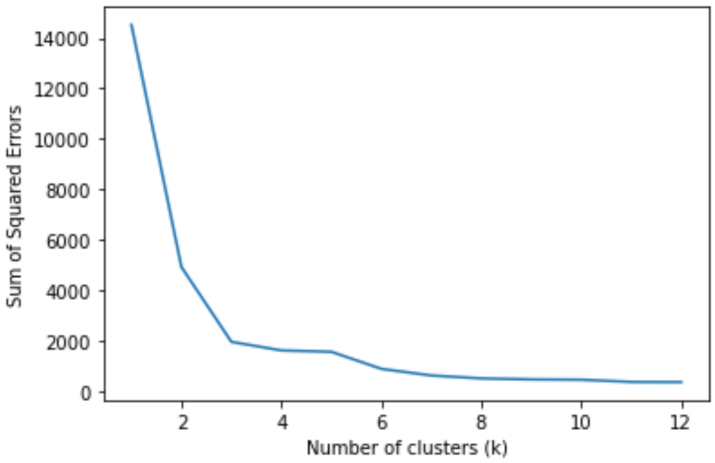




If useElbowMethod flag is set to True, then kmeans algorithm is executed for all values of k ranging between given lower and upper bounds on k (low and high). For each value of k, sum of squared errors is computed and stored. It then plots a line chart with number of clusters (k) on x-axis and sum of squared errors (SSE) on y-axis and asks for the value of k at elbow point as input. Once the value of k at elbow point is given as input, it executes k-means clustering algorithm and returns centroids and cluster assignments.

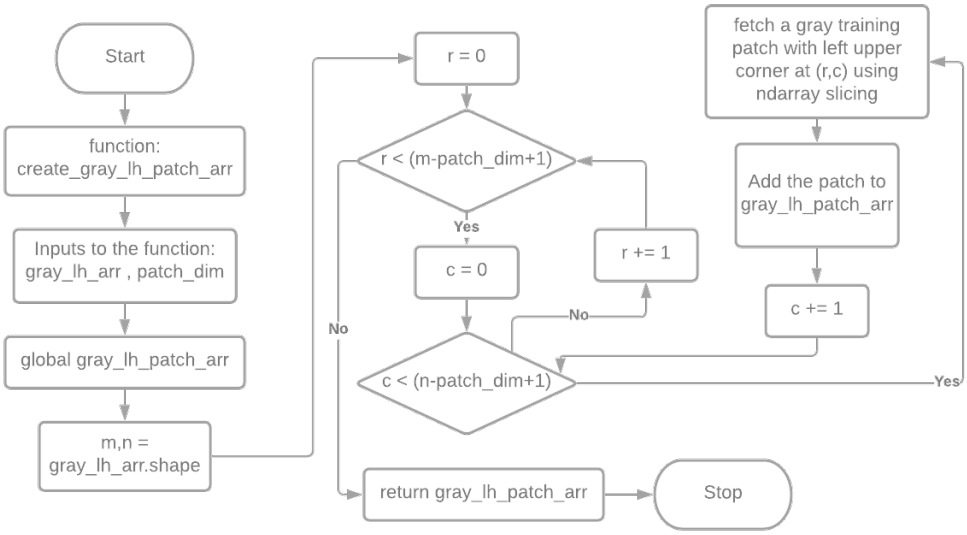


Sample plot of elbow data:

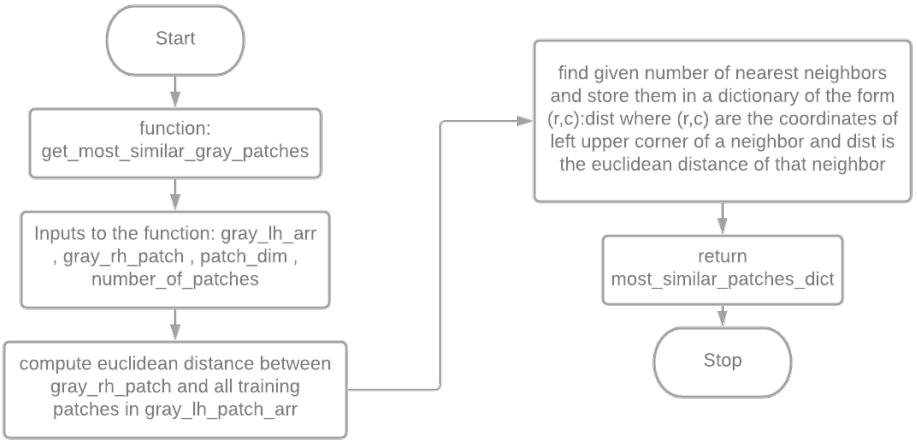


3.3 Coloration of right half of an image by Basic Agent

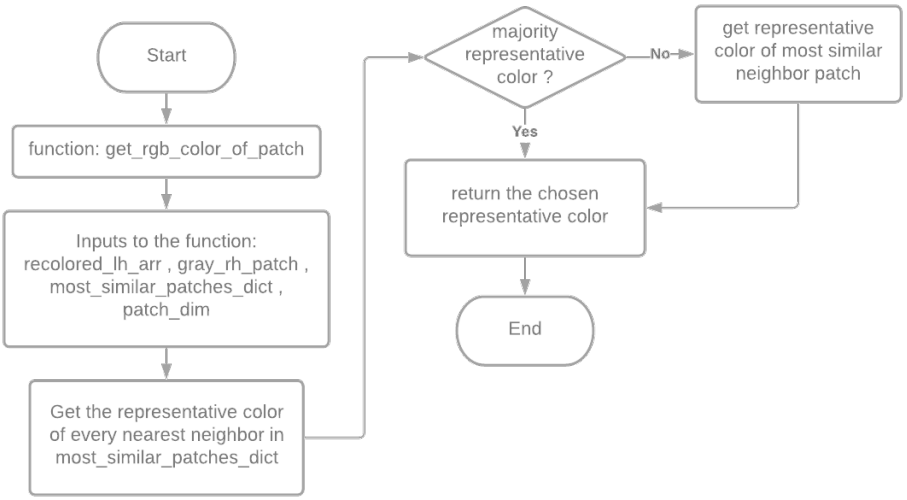
Left and right halves of gray-scaled image and recolored left half of colored image are given as input to the basic AI agent. As the recolored left half has limited number of colors, AI agent treats coloration of gray-scaled right half as a classification task. Gray-scaled left half and corresponding recolored left half form the training set of AI agent. Gray-scaled right half forms the test set of AI agent. Basic agent uses K-nearest neighbors algorithm to do the classification. It is used as an instance-based learning algorithm. Agent uses 3 x 3 patch in a gray-scaled image as the collection of features for a pixel in the middle of a patch. For every such patch in the test set, it finds 6 nearest neighbors (3 x 3 patches) in the training set. Euclidean distance between test patch and training patch is used as a similarity measure to find the nearest neighbors. For all 6 nearest neighbors, AI agent already has corresponding RGB color of the middle pixels in the recolored left half. If there is a majority color in the 6 representative colors, it gets assigned to the middle pixel of a test patch. If there is no majority color in the 6 representative colors, then the representative color of the nearest (most similar) training patch gets assigned to the middle pixel of a test patch. Agent colors the middle pixels of all possible 3 x 3 patches in test set in above-mentioned fashion. As the boundary pixels in the test set are not the middle pixels of any 3 x 3 patch, they are assigned black color (r=0, g=0, b=0).
Since basic agent has to compute euclidean distance of a test patch with all possible training patches to find the nearest neighbors, agent creates ndarray of all possible training patches to compute euclidean distances in a vectorized manner.

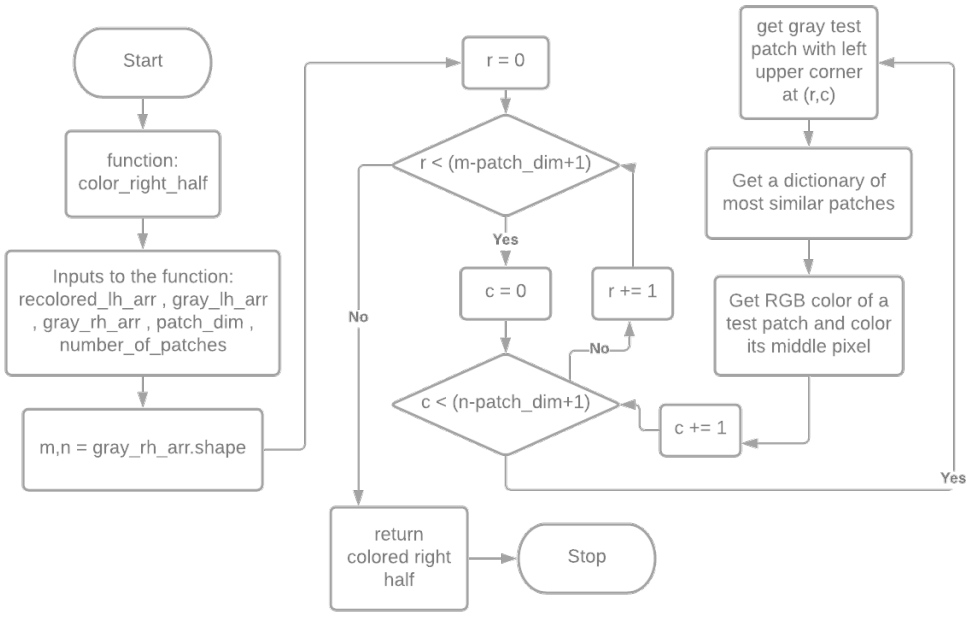


Agent computes euclidean distance of a test patch with all training patches in a vectorized manner. It then finds the given number of nearest neighbors based on computed distances. These neighbors are stored in a dictionary where key represents location of left upper corner of neighbor patch and value represents the euclidean distance with that neighbor patch.



Once the information of nearest neighbors is stored in a dictionary, agent has to use it to color the middle pixel of a test patch. It takes RGB color of middle pixels of nearest neighbor patches from the recolored left half as representative colors. If there is a majority representative color, then the middle pixel of a test patch is colored with this representative color. If there is no majority representative color, then the middle pixel of a test patch is colored with the RGB representative color of most similar patch.

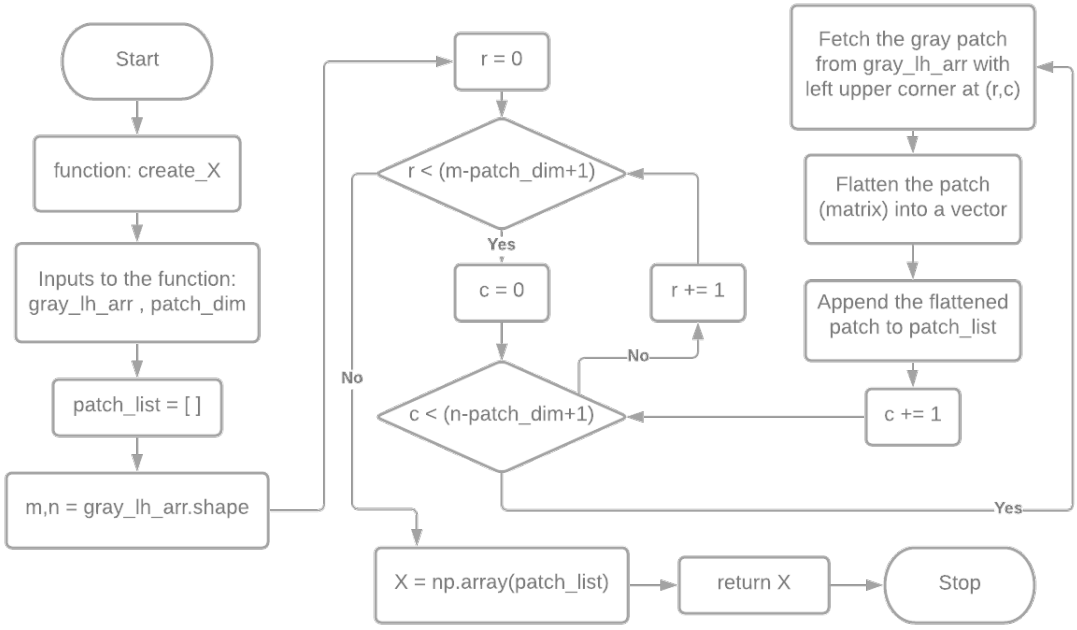




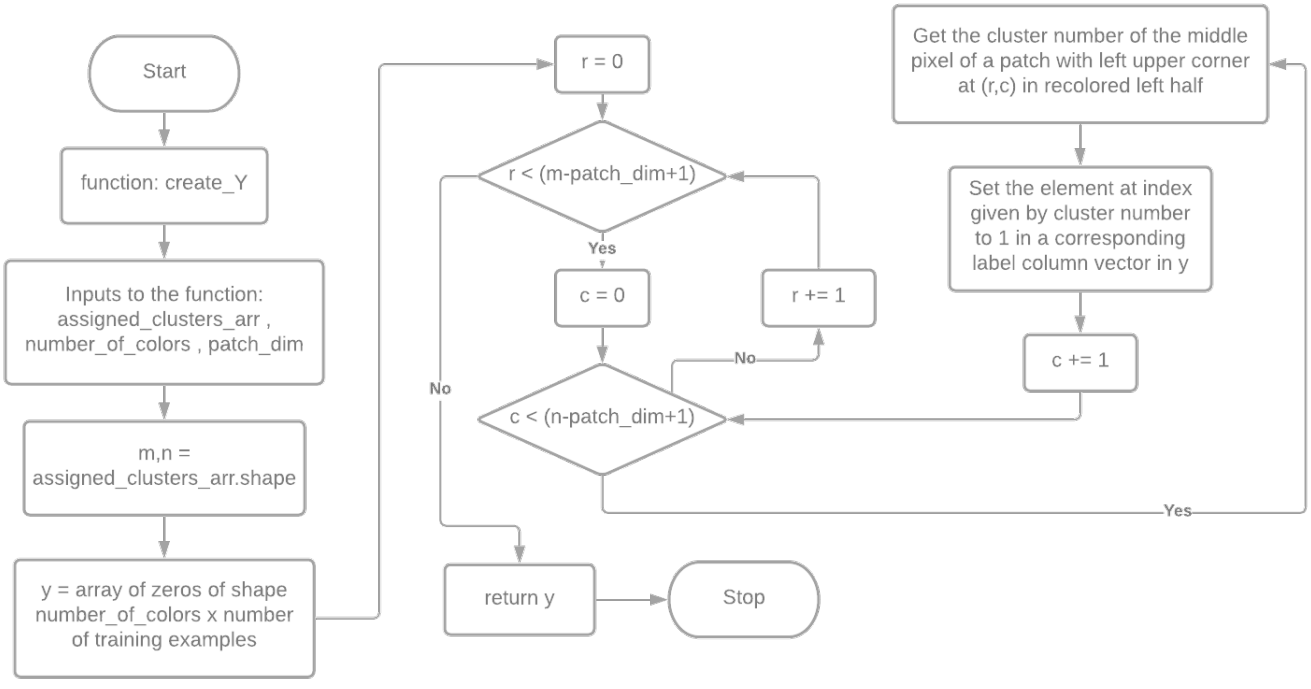
Flexible parameters in basic AI agent:
1) Dimension of a patch.
2) Number of colors (k) in k-means clustering
3) Number of nearest neighbors in KNN.

3.4 COLORATION OF RIGHT HALF OF AN IMAGE BY IMPROVED AGENT

Left and right halves of gray-scaled image and recolored left half of colored image are given as input to the improved AI agent. As the recolored left half has limited number of colors, AI agent treats coloration of gray-scaled right half as a classification task. Gray-scaled left half and corresponding recolored left half form the training set of AI agent. Gray-scaled right half forms the test set of AI agent. Improved AI agent uses feedforward neural network with sigmoid activation function to do the soft classification.
Agent uses patch_dim x patch_dim patch in a gray-scaled image as a collection of features for a pixel in the middle of a patch. It creates a matrix of flattened feature patches as the feature matrix to be used for training a neural network.



As the agent is treating coloration of gray-scaled right half as a classification task, it also needs labels corresponding to feature matrix to train a neural network (supervised learning). Agent uses one-hot encoding in the output layer to perform binary or multi-class classification. If there are k colors in the recolored left half (output of k-means clustering module), then output layer of a neural network has k neurons with one-hot encoding.

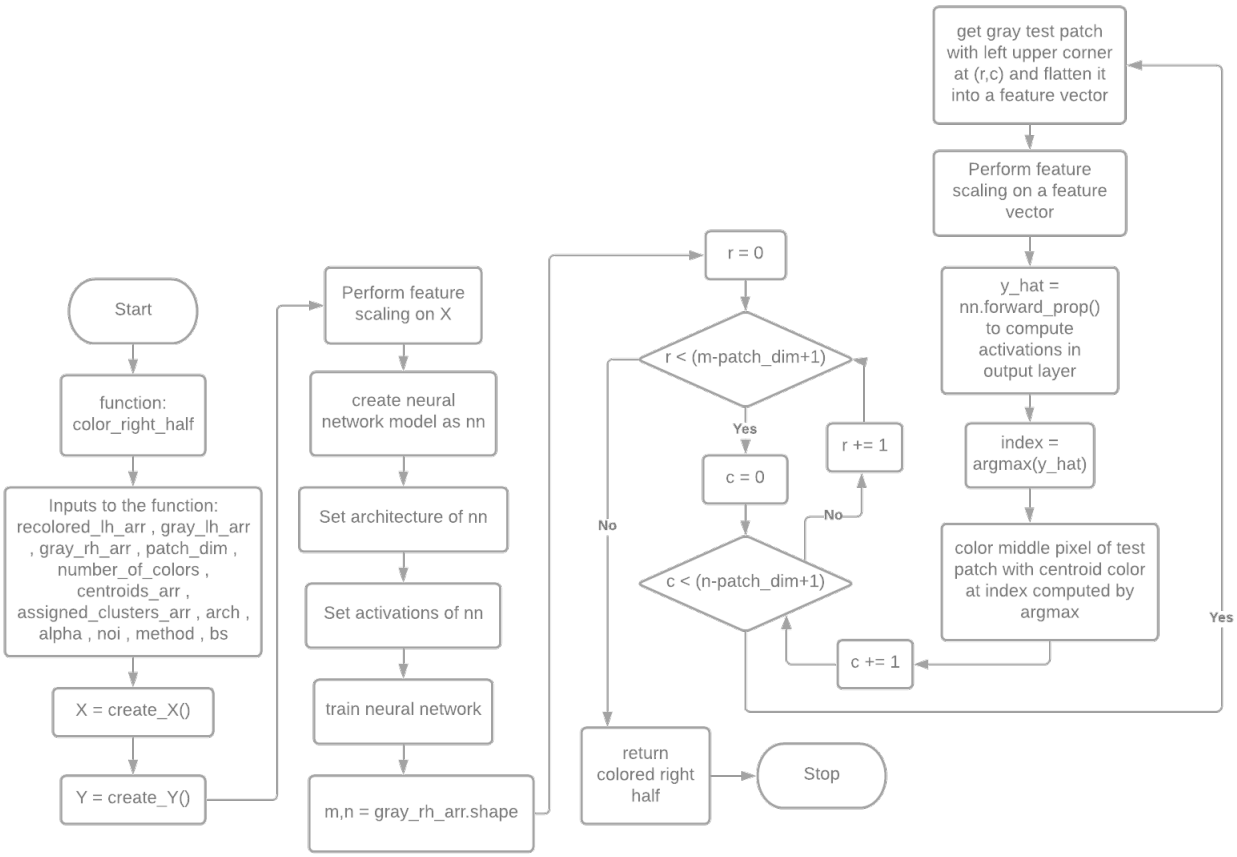


Improved agent also performs feature scaling on a feature matrix X by using mean normalization.

Mean normalization

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

Once the feature matrix X and label matrix Y are ready and feature scaling is performed, agent creates neural network model as an object of class NeuralNetwork. Agent sets the architecture of a neural network by passing an architecture list as an argument to set_architecture method. Agent also sets the activation functions of one or more hidden layers and output layer by passing a list of activation strings ('sigm','relu') to set_activations method. It then trains a neural network by using backpropagation algorithm. Trained neural network is used by the agent to color gray-scaled right half.



Structure of class Neural Network:

- def __init__(self,w_list=None,b_list=None):
- def set_architecture(self,l):
- def set_activations(self,l):
- def initialize_parameters(self):
- def forward_prop(self,X):
- def compute_cost(self,A,Y):
- def train(self,X,Y,alpha,noi,method='sgd',bs=1):

Improved agent has flexibility in terms of architecture of a neural network as well as activations in different hidden layers and output layer of a neural network.

set_architecture method takes a list of integers as input to set the neural network architecture. A list consists of number of neurons in input layer, followed by number of neurons in one or more hidden layers and number of neurons in the output layer. All the integers represent number of neurons excluding the bias unit. Bias unit is added in the input layer and all hidden layers by the neural network automatically.

set_activations method takes a list of activation strings as input to set the activations of all hidden layers and output layer. Possible activation strings are 'sigm' (sigmoid) and 'relu' (rectified linear unit).

initialize_parameters method randomly initializes all the weight matrices. Weight matrices excluding weights of bias units are stored in w_list. Weights of bias units are stored in bias_list.

forward_prop method takes feature matrix as input and computes activations in all hidden layers and output layer for all the examples in feature matrix. All the computed activations are stored in A_list. Forward propagation is done in a vectorized manner to speedup the process.

train method uses backpropagation algorithm to minimize the cross entropy loss and iteratively update the parameters of a neural network. Agent can use batch gradient descent ('gd'), mini-batch gradient descent ('mbgd') or stochastic gradient descent ('sgd') by passing corresponding method string. If mini-batch gradient descent is to be used, batch size can be passed as an argument to train method. Learning rate and number of iterations are also passed as arguments to the train method.

Once the neural network is trained, its updated parameters are used by color_right_half function to color the gray-scaled right half.

Flexible parameters in improved AI agent:
1) Dimension of a patch
2) Number of colors (k) in k-means clustering
3) Architecture of feedforward neural network (number of neurons in the input layer , number of neurons in the output layer , number of hidden layers , number of neurons in each hidden layer)
4) optimization method
5) learning rate
6) activation functions

4 Basic Agent (KNN) Results

4.1 Specification of Basic Agent

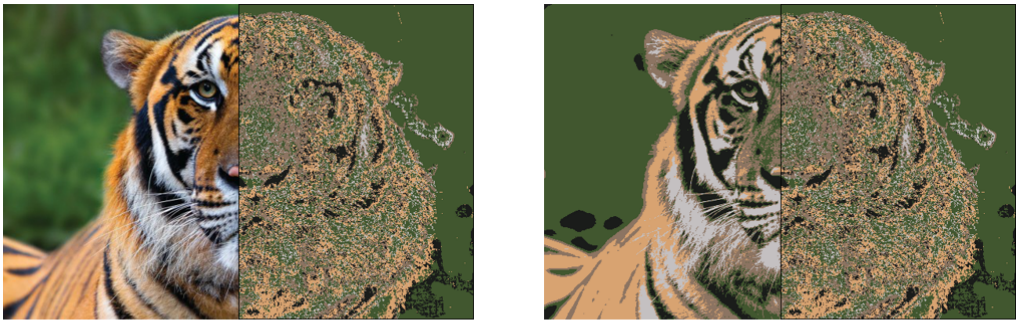
Dimension of a patch: 3
Number of colors (k) in k-means clustering: 5
Number of nearest neighbors in KNN: 6
Similarity measure in KNN: euclidean distance

Original Image:
It is a 3-dimensional matrix of integers varying from 0 to 255.

Input Space:
Gray-scaled left and right halves as well as left half recolored by k-means are given as input to the basic AI agent. Gray-scaled left and right halves are the 2-dimensional matrices of integers varying from 0 to 255. Recolored left half is a 3-dimensional matrix of integers varying from 0 to 255. These matrices form the **input space** of AI agent.

Recolored Left Half (k = 5) Gray-scaled left half Gray-scaled right half

Output Space:
Basic agent colors gray-scaled right half by using input matrices as the training set and outputs colored right half. Colored right half is a 3-dimensional matrix of integers varying from 0 to 255. These matrices form the output space of AI agent.



Original Left Half Right Half colored by KNN Left Half colored by k-means Right Half colored by KNN

Model Space:
Basic agent uses k-nearest neighbors algorithm to perform the hard classification. k-nearest neighbors algorithm with varying values of integer k and different similarity measures form the model space of AI agent.
Since basic agent uses k-nearest neighbors algorithm as instance-based machine learning, it does not minimize the cost function by any learning algorithm.

4.2 Parameters of the model

Parameters of the basic AI agent are same as given in the problem statement.
Agent uses number of colors (k) = 5 in k-means clustering algorithm.
Number of nearest neighbors used by AI agent in KNN is 6.
KNN algorithm requires similarity measure to find the nearest neighbors.
Agent uses euclidean distance as the measure of similarity. (Lesser distance implies more similarity)
Euclidean distance is chosen because it represents the geometric distance between two data points in metric space and satisfies the properties of metric space.

4.3 Qualitative and Quantitative evaluation of the output

1) Training error when compared with original image

Original Left Half

Left Half colored by KNN

Since there are more than a thousand different colors in the original left half and only 5 colors in the left half colored by KNN, MSE, RMSE and MAD are computed instead of classification metrics.

MSE = 253.9155
RMSE = 15.9347
MAD = 391.0248
Even though it is visually possible to identify a tiger in the left half colored by KNN, it does not have a smooth colorization.

2) Training error when compared with image recolored by k-means.

Left Half colored by k-means

Left Half colored by KNN

Since there are only 5 colors (number of classes) in left half colored by k-means and left half colored by KNN, classification metrics can yield fair comparison.

Confusion Matrix:

Predicted Class

Actual Class

	7996	157	985	514	1051
	793	51347	2351	3616	2463
	1969	1120	15658	2738	995
	183	581	249	5399	512
	370	497	405	733	15922

Non-zero entries in positions except main diagonal indicate misclassifications of basic agent.

Class	Precision	Recall	F1-score	Support
	0.7069	0.7471	0.7264	10703
	0.9561	0.8477	0.8986	60570
	0.7969	0.6965	0.7433	22480
	0.4153	0.7798	0.542	6924
	0.7603	0.8882	0.8193	17927

3) Testing error when compared with original image

Original Right Half

Right Half colored by KNN

Since there are more than thousand different colors in the original right half and only 6 colors in the right half colored by KNN, MSE, RMSE and MAD are computed instead of classification metrics.

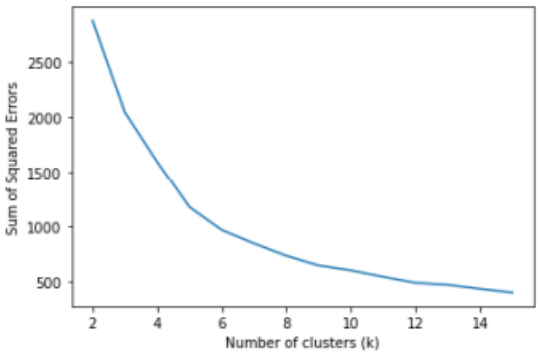
MSE = 295.3502
RMSE = 17.1858
MAD = 353.1999

Colorization of the gray-scaled right half (test image) is neither visually nor numerically satisfying as compared to the colorization of train image.

4.4 POSSIBLE IMPROVEMENTS TO THE KNN MODEL

Training set (image) for KNN algorithm consists of image recolored by k-means clustering algorithm. Training image created with a larger value of k will be a better representative of the original image. Also, determining an optimal value of k (elbow method, silhouette method) will yield better coloration results.
Hyperparameter in KNN algorithm is the number of nearest neighbors. Grid search over a range of values will help find an optimal value of k (number of nearest neighbors).
Basic agent implemented KNN as instance-based machine learning that requires a time-consuming scan through an entire dataset. Use of space-partitioning data structure such as k-d tree will speed-up the algorithm.

For an image of a tiger that is used by AI agents, k = 8 appears to be the optimal value according to elbow method.



5 Improved Agent (Shallow Neural Network) Results

5.1 Specification of Shallow Neural Network

Dimension of a patch: 3
Number of colors (k) in k-means clustering: 8
Number of neurons in the input layer: 9 inputs + 1 bias
Number of neurons in the output layer: 8
Number of hidden layers: 2
Number of neurons in hidden layer 1: 20 inputs + 1 bias
Number of neurons in hidden layer 2: 20 inputs + 1 bias
Activation function in both hidden layers and output layer: Sigmoid Function
Optimization method: Batch Gradient Descent
Learning rate: 0.00002
Number of iterations: 10000

Original Image:
It is a 3-dimensional matrix of integers varying from 0 to 255.

Input Space:
Gray-scaled left and right halves as well as left half recolored by k-means are given as input to the improved AI agent. Gray-scaled left and right halves are the 2-dimensional matrices of integers varying from 0 to 255. Recolored left half is a 3-dimensional matrix of integers varying from 0 to 255. These matrices form the **input space** of AI agent.

Recolored Left Half (k = 8)

Gray-scaled left half

Gray-scaled right half

Output Space:
Improved agent colors gray-scaled right half by using input matrices as the training set and outputs colored right half. Colored right half is a 3-dimensional matrix of integers varying from 0 to 255. These matrices form the **output space** of AI agent.

Original Left Half | Right Half colored by Shallow Neural Network | Left Half colored by k-means (k = 8) | Right Half colored by Shallow Neural Network

Model Space:
Improved agent uses shallow neural network with sigmoid activation function in the output layer to perform the soft classification. Neural network with varying architectures, activation functions and values of parameters (weights) form the **model space** of AI agent.
As neural network performs classification task, cross entropy loss is used as the cost function (error function).
Backpropagation algorithm with batch gradient descent is used as the learning algorithm (optimization method) to minimize cross entropy loss by updating the parameters (weights).

5.2 Choice of parameters of the model

As neural network uses pixels in 3 x 3 patch as the features of a middle pixel, it has 9 neurons in the input layer. 1 bias unit is added to the input layer to allow translation.
As neural network uses 8 colors (k=8) in k-means clustering and one-hot encoding in the y labels, it has 8 neurons in the output layer. Since it performs soft classification task, it uses sigmoid activation function in the output layer to map the outputs in a range (0,1) that can be interpreted as class-conditional probabilities.
Neural network uses non-linearity (sigmoid function) in the hidden layers to allow approximation of any function.
According to Universal Approximation Theorem, a feedforward neural network with a single hidden layer is sufficient to approximate any function, but the hidden layer can be infeasibly large and may fail to learn and generalize correctly.
To resolve these issues, two feasibly large hidden layers are used by the agent to allow approximation of any function as well as generalization.
Choice of sigmoid activation function in hidden layers over relu is driven by the empirical observations of the convergence results.
Instead of mini-batch and stochastic gradient descent, batch gradient descent is used as an optimization method for consistent minimization of the cost function (error function).
Multiple trial and errors with different learning rates led to a choice of 0.00002 as the learning rate of batch gradient descent.

5.3 Preprocessing of the input and output data

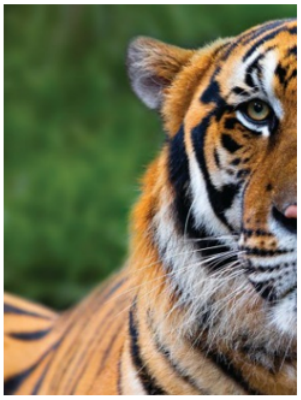
Since original image has a wide variety of colors (classes), agent performs k-means clustering on the colored left half to narrow down the number of colors (classes) in the training set.
3 x 3 patch in the gray-scaled left half is flattened into a column vector to serve as the feature vector of middle pixel of a patch.
Since this is a multi-class classification problem and classes do not posses any order (RGB color is a nominal attribute), agent performs one-hot encoding on the true labels (centroid color assignments).
Feature scaling is performed on a feature matrix by using mean normalization as it improves convergence speed of gradient descent.

5.4 Training of the model

Agent has 3 optimization methods to use for training a neural network : batch gradient descent ('gd') , mini-batch gradient descent ('mbgd') , stochastic gradient descent ('sgd').
Agent used batch gradient descent as an optimization method for its faster convergence rate. Mini-batch gradient descent and stochastic gradient descent have slower convergence rates as they use estimations of gradients.

5.5 Qualitative and Quantitative evaluation of the output

1) Training error when compared with original image



Original Left Half

Left Half colored by Shallow Neural Network

Since there are more than thousand different colors in the original left half and only 8 colors in the left half colored by neural network, MSE, RMSE and MAD are computed instead of classification metrics.

MSE = 51.8109
RMSE = 7.1979
MAD = 88.5064

Colorization performed by improved agent is numerically as well as visually better as compared to basic agent. Improved agent's colorization is much smoother than that of basic agent.

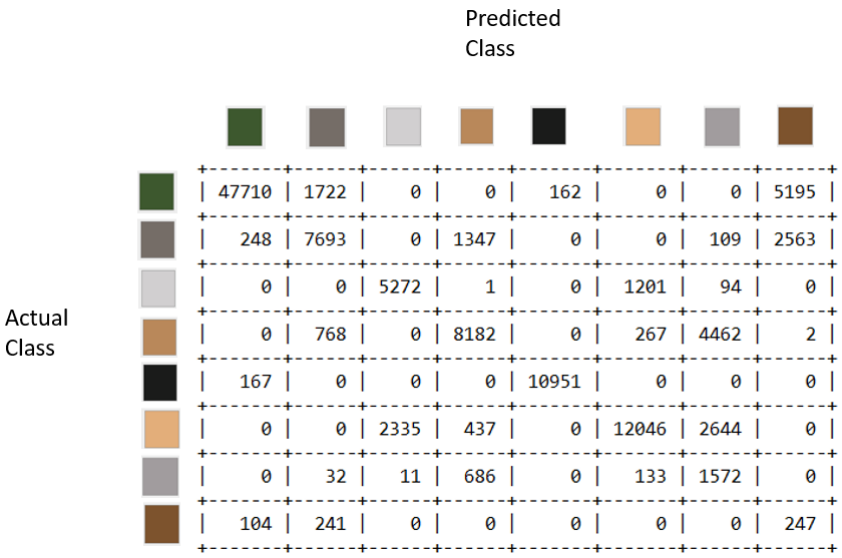
2) Training error when compared with image recolored by k-means.

Left half colored by k-means (k = 8)

Left Half colored by Shallow Neural Network

Since there are only 8 colors (number of classes) in left half colored by k-means and left half colored by neural network, classification metrics can yield fair comparison.

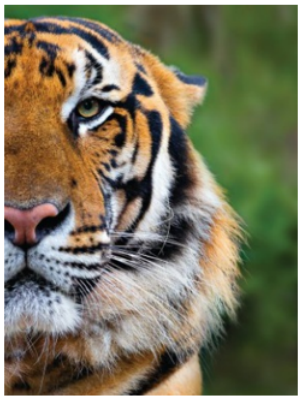
Confusion Matrix:



Higher values in main diagonal entries and a lot of 0 entries at other places indicate better classification capabilities of improved agent.

Class	Precision	Recall	F1-score	Support
	0.9892	0.8708	0.9262	54789
	0.7357	0.6432	0.6863	11960
	0.692	0.8027	0.7433	6568
	0.768	0.5981	0.6725	13681
	0.9854	0.985	0.9852	11118
	0.8827	0.6898	0.7744	17462
	0.177	0.6459	0.2779	2434
	0.0308	0.4172	0.0574	592

3) Testing error when compared with original image



Original Right Half

Right Half colored by Shallow Neural Network

Since there are more than thousand different colors in the original right half and only 8 colors in the right half colored by neural network, MSE, RMSE and MAD are computed instead of classification metrics.

MSE = 245.7721
RMSE = 15.6771
MAD = 310.7566

Similar to training image, improved agent's colorization of test image is also visually and numerically better than that of basic agent.

5.6 Possible improvements to the Shallow Neural Network

Improved agent uses 3 x 3 patch (9 pixels) as features of the middle pixel. Increasing the patch dimension (and hence number of features) will allow better pattern recognition.
Increasing number of hidden layers as well as number of neurons in those hidden layers (making deep neural network) will better capture the complex pattern in the data.
Use of advanced optimization methods such as adagrad and rmsprop will yield faster and better convergence as compared to gradient descent.
Use of learning rate schedules will also yield faster and better convergence.
Use of advanced neural network architectures such as convolutional neural networks (Convolution layers + max pooling layers + fully connected layers) can lead to better colorization of gray-scaled images.

6 AGENT COMPARISONS

1) Training error when compared with original image.

Metric	Basic Agent	Improved Agent
MSE	253.9155	51.8109
RMSE	15.9347	7.1979
MAD	391.0248	88.5064

2) Testing error when compared with original image.

Metric	Basic Agent	Improved Agent
MSE	295.3502	245.7721
RMSE	17.1858	15.6771
MAD	353.1999	310.7566

7 CONCLUSION

Basic AI agent does the colorization by using KNN algorithm. Since the dataset is higher dimensional, KNN faces the curse of dimensionality. Colorization results of KNN are numerically as well as visually worse when compared with the improved AI agent.
Improved AI agent does the colorization by using shallow neural network. Neural network with non-linear activation functions is better able to capture the complex pattern in the dataset. Colorization results of Neural network are numerically as well as visually much better as that of basic agent. Also, neural networks are less impacted by the curse of dimensionality when compared with K-nearest neighbors algorithm.

8 REFERENCES

1) https://en.wikipedia.org/wiki/K-means_clustering
2) https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
3) https://en.wikipedia.org/wiki/Feedforward_neural_network
4) https://en.wikipedia.org/wiki/Gradient_descent
5) https://www.oreilly.com/library/view/learning-python-5th/9781449355722/
6) https://pillow.readthedocs.io/en/stable/
7) https://matplotlib.org/