```
%==========================================================================
==============
%LOADS TRAINING DATASET
%==========================================================================
==============

clear

pkg load image

directory_path = '/Users/charlesdavi/Desktop/MNIST/MNIST - JPG - training/';

counter = 1;

for k = 0 : 9

  num_images = 500;

  category_number = k;

  directory = [directory_path int2str(k) '/'];

  for i = 1 : num_images

    I = imread([directory '1 (' int2str(i) ').jpg']);

    MNIST_array{counter} = I;
    MNIST_category{counter} = k; %stores the classifier associated with the image
    counter = counter + 1;

  endfor

endfor

%==========================================================================
==============
%EXTRACTS SHAPE INFORMATION
%==========================================================================
==============
tic;
[final_avg_matrix final_indexes] = partition_image_vectorized_gs(MNIST_array{1});
%this is to size the partitions for the entire dataset
toc
N = size(final_avg_matrix,1);

tic;
total_num_images = 10*num_images; %there are 10 categories of images
```

```
corresponding to the digits 0 through 9

counter = 1; %index for the observations
dataset = [];

scramble = randperm(total_num_images,total_num_images); %this is to permute the
dataset, which would otherwise be sorted by digit

%iterates through entire dataset
while(counter <= total_num_images)

  I = MNIST_array{scramble(counter)};

  [avg_matrix] = calc_avg_color_vect(final_indexes, I, N); %this extracts shape
information

  input_vector = reshape(avg_matrix, [1 N^2]);

  input_vector(N^2+1) = MNIST_category{scramble(counter)}; %this is the hidden
classifier

  input_vector(N^2+2) = scramble(counter); %this is the hidden classifier

  dataset(counter,:) = input_vector;

  counter = counter + 1;

endwhile
toc

%================================================================
%LOADS TESTING DATASET
%================================================================

directory_path = '/Users/charlesdavi/Desktop/MNIST/MNIST - JPG - training/';

counter = 1;

clear MNIST_array
clear MNIST_category %stores the classifier associated with the image

for k = 0 : 9

  category_number = k;

  directory = [directory_path int2str(k) '/'];
```

```
  for i = num_images + 1 : 2*num_images %uses the remaning images in the training
dataset

    I = imread([directory '1 (' int2str(i) ').jpg']);

    MNIST_array{counter} = I;
    MNIST_category{counter} = k; %stores the classifier associated with the image
    counter = counter + 1;

  endfor

endfor

%================================================================
==============
%EXTRACTS SHAPE INFORMATION
%================================================================
==============
total_num_images = 10*num_images; %there are 10 categories of images
corresponding to the digits 0 through 9

counter = 1; %index for the observations
testing_dataset = [];

scramble = randperm(total_num_images,total_num_images); %this is to permute the
dataset, which would otherwise be sorted by digit

%iterates through entire dataset
while(counter <= total_num_images)

  I = MNIST_array{scramble(counter)};

  [avg_matrix] = calc_avg_color_vect(final_indexes, I, N); %this extracts shape
information

  input_vector = reshape(avg_matrix, [1 N^2]);

  input_vector(N^2+1) = MNIST_category{scramble(counter)}; %this is the hidden
classifier

  input_vector(N^2+2) = scramble(counter); %this is the hidden classifier

  testing_dataset(counter,:) = input_vector;

  counter = counter + 1;

endwhile
```

```
%================================================================
%GENERATES SUPERVISED VALUES OF DELTA
%================================================================

N = N^2; %housekeeping, don't do twice

num_rows = size(dataset,1);
cluster_matrix = zeros(num_rows,num_rows);

tic;
for i = 1 : num_rows

  [delta final_rows] = supervised_delta(dataset, i, N);

  delta_vector(i) = delta;

  if(final_rows != 0)

  cluster_matrix(i,final_rows) = 1;

  endif

endfor
toc

%================================================================
%MAKES PREDICTIONS; TESTS ACCURACY
%================================================================

num_errors = 0;
num_rejections = 0;

tic;
for i = 1 : num_rows

  input_vector = testing_dataset(i,:);
  actual_class = input_vector(N+1);

  [predicted_vector predicted_class prediction_row] = find_NN(input_vector, dataset,N);

  if(norm(predicted_vector - input_vector) > delta_vector(prediction_row)) %if true, then
the prediction is rejected

    num_rejections = num_rejections + 1;

  elseif(predicted_class != actual_class) %if true, then the allowed prediction is an error
```

```
    num_errors = num_errors + 1;

  endif

endfor
toc

accuracy = 1 - num_errors / (num_rows - num_rejections)
```