```
%=====================================================================
==========
%=====================================================================
==========
%BLACK TREE VECTORIZED DEEP LEARNING
%=====================================================================
==========
%=====================================================================
==========
%COPYRIGHT CHARLES DAVI

%=====================================================================
==========
%RUNS IMAGE CLASSIFICATION
%=====================================================================
==========

clear all
clc

dataset_file = "/Users/charlesdavi/Desktop/Datasets/
MNIST_Fashion_Full/img/fashion0.png";

[root prefix extension initial_index] =
get_file_components_BlackTree(dataset_file);
num_images = 15000;

%assumes sequential file names
for i = 1 : num_images

  I = imread([prefix, int2str(initial_index + i - 1), extension]);

  IMG_array{i} = I;

  %tests for gray scale dimensions
  x = size(I); %the dimensions of the image
  y = size(x,2); %the number of columns in the image

endfor

%loads
classifiers-----------------------------------------------------------
---

  file_name = [root, 'class_vector.txt'];
  A = csvread(file_name);

  IMG_category = A(1:num_images);

%=====================================================================
==========
%EXTRACTS SHAPE INFORMATION
%======================================================================
==========
tic;

I_sample = IMG_array{1};
[final_avg_matrix final_indexes] =
partition_image_vectorized_gs_BlackTree(I_sample); %this is to size
the partitions for the entire dataset

orig_im_size = size(I_sample,1)*size(I_sample,2); %this is the orignal
total image size

N = size(final_avg_matrix,1);

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

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

  I = IMG_array{counter};

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

  input_vector = reshape(avg_matrix, [1 N^2]);
  input_vector(N^2+1) = IMG_category(counter); %this is the hidden
classifier

  dataset(counter,:) = input_vector;

  counter = counter + 1;

endwhile

im_proc_time = toc;
num_rows = size(dataset,1);

%======================================================================
==========
%RUNS PREDICTION
%======================================================================
==========

num_rows = size(dataset,1);
N = size(dataset,2) - 1;

%copies classifiers and generates training / testing datasets
```

```
dataset(:,N+2) = dataset(:,N+1);
testing_percentage = .15;
num_testing_rows = floor(num_rows*testing_percentage);
testing_rows = randperm(num_rows,num_testing_rows);
dataset(testing_rows,N+1) = -1; %flags testing rows

%sorts dataset
tic;
sorted_dataset = sortrows(dataset,1:N);

%finds the testing rows in the sorted dataset
sorted_testing_rows = find(dataset(:,N+1) == -1);

%applies prediction
for i = 1 : num_testing_rows

  testing_row = sorted_testing_rows(i);

  %right hand side---------------------------------------------------

  %if true, then it's not the end of the list
  if(testing_row != num_rows)

    j = testing_row + 1;
    RH_class(i) = sorted_dataset(j,N+1);

    %if true, then the entry is a training row
    if(RH_class(i) != -1)

      RH_cluster_size(i) = 1;
      break_loop = 0;

    %otherwise, the entry is a testing row
    else

      RH_cluster_size(i) = 0;
      break_loop = 1;

    endif

    %finds the cluster until first error
    while(break_loop == 0 && j <= num_rows)

      test_class = sorted_dataset(j,N+1);

      %if true, then we increment the cluster size
      if(test_class == RH_class(i))

        RH_cluster_size(i) = RH_cluster_size(i) + 1;
      %otherwise, we break the loop
      else

        break_loop = 1;

      endif

    j = j + 1;

    endwhile

  %otherwise, it's the end of the list and so we flag a rejection
  else

    RH_cluster_size(i) = 0;
    RH_class(i) = -1;

  endif

  %left hand side----------------------------------------------------

  %if true, then it's not the beginning of the list
  if(testing_row != 1)

    j = testing_row - 1;
    LH_class(i) = sorted_dataset(j,N+1);

    %if true, then the entry is a training row
    if(LH_class(i) != -1)

      LH_cluster_size(i) = 1;
      break_loop = 0;

    %otherwise, the entry is a testing row
    else

      LH_cluster_size(i) = 0;
      break_loop = 1;

    endif

    %finds the cluster until first error
    while(break_loop == 0 && j > 0)

      test_class = sorted_dataset(j,N+1);

      %if true, then we increment the cluster size
      if(test_class == RH_class(i))

        LH_cluster_size(i) = LH_cluster_size(i) + 1;
      %otherwise, we break the loop
      else

        break_loop = 1;

      endif

      j = j - 1;

    endwhile

  %otherwise, it's the beginning of the list and so we flag a
rejection
  else

    LH_class(i) = -1
    LH_cluster_size(i) = 0;

  endif

endfor
toc

%tests accuracy------------------------------------------------------

tic;
num_errors = 0;
num_rejections = 0;

error_vector = zeros(1,num_testing_rows);

for i = 1 : num_testing_rows

  testing_row = sorted_testing_rows(i);
  actual_class = sorted_dataset(testing_row,N+2)

  %righthand prediction
  RH_temp_cluster_size = RH_cluster_size(i);
  RH_prediction_vector = RH_class(i)*ones(1,RH_temp_cluster_size);
%creates a vector with class labels

  %lefthand prediction
  LH_temp_cluster_size = LH_cluster_size(i);
  LH_prediction_vector = LH_class(i)*ones(1,LH_temp_cluster_size);
%creates a vector with class labels

  %tests prediction accuracy

  prediction_vector = [RH_prediction_vector LH_prediction_vector]
  %if true, then the cluster is empty, or the classes are not equal, a
rejection
  if((size(prediction_vector,2)) == 0 || (LH_class(i) != RH_class(i)))

    num_rejections = num_rejections + 1;
    predicted_class_vector(i) = -1;

  %otherwise, we find the modal class
  else

    predicted_class = mode(prediction_vector)
    predicted_class_vector(i) = predicted_class;

    %if true, the prediction is an error
    if(predicted_class != actual_class)

      num_errors = num_errors + 1;
      error_vector(i) = 1;

    endif

  endif

endfor

toc
accuracy = 1 - num_errors/(num_testing_rows - num_rejections)

%applies confidence--------------------------------------------------

cluster_size_vector = RH_cluster_size .+ LH_cluster_size;
max_cluster_size = max(cluster_size_vector)
actual_class_vector = sorted_dataset(sorted_testing_rows,N+2);

for i = 1 : max_cluster_size

  x = find(cluster_size_vector >= i);
  num_predictions = size(x,2);

  num_errors = sum(error_vector(x));

  y = find(predicted_class_vector(x) == -1);
  num_rejections = size(y,2);

  accuracy(i) = 1 - num_errors/(num_predictions - num_rejections);

endfor

figure, plot(accuracy)
```