Date: 14th Aug 2017

Image Recognition and Retrieval

(Using KNN Technique)

Author: Harshal Vijay Neelkamal

Introduction:

This report is part of my final project for INFO 6205(Data Structures and Algorithms). It contains brief explanation and demo directions for working of my Java swing application for image recognition and retrieval. The application is configured to only detect fruits and the training set should be composed of images having white or transparent backgrounds.

Research and Findings:

1. ***KNN:***
   KNN is one of the few machine learning techniques that I stumbled upon while researching for this project. The idea is, to maintain a base of information extracted from your learning sample in a K-Dimensional space and give out the predictions based on the proximity of test sample to Nodes in same K-Dimensional space.

2. ***K-Mean:***
   K-Mean is another machine learning technique which is effectively a subtype of KNN technique. Here, the idea is to find the centre positions for different clusters of data and then make a n enclosure or a pocket anything outside of which is negative to that cluster and anything inside is a potential part of that cluster.

3. ***K-D Tree:***
   K-D tree is one of those data structures that bestr suits the storage of nodes in K-Dimensional space. I have used this very data structure in my algorithm.

4. ***Eigen Faces:***
   Eigen faces is one other concept that I researched on and then I dropped it of since it didn't involve much of data structures. It's an effective technique in face or object detection with certain limitations. We create an Average face and subtract it from all other training sets to produce varied points in a 2-dimensional space. Then using Principal variance component analysis, change the axis of variance for the image and thus we extract something called as eigen faces.

For detection, we calculate the eigen distance of all pixels of text image with all pixels of all the eigen faces (training images) and the test image is then classified as the label to the closest image is the training set.

5. ***Feature Extraction:***

Feature extraction forms the most important part of this project and each feature acts as a dimension in K dimensional space created for KNN.

Processing is taking place on three fronts for feature extraction in this project.

1.) Mean HSV values of all the all the pixels in an image serve as 3 of the 4 dimensions in one of the trees.

2.) The mean luma of the image serves as 1 of 4 dimensions in the same tree.

3.) The ratios of left and right diagonals to the height and the width, which is dependent on shape of the fruit, serve as 4 dimensions for 2^{nd} tree.

A combination of results from the 2 trees is used to make predictions.

Data Structures & Algorithms Used

1. **K-Dimensional Tree:**
    - For training purpose, there is need of some data structure which can hold data and arrange it based on the features extracted.
    - 2 trees are used in this project for 4 features each, making it 2 4-Dimensional trees.
    - A K dimensional tree gives a virtual effect of storing the data in K dimensional space such that all the neighbours in K dimension are clustered together under a single parent node in this K-dimensional tree.
    - Doing so, makes it easier to find N-Nearest neighbours.

- All you need to do is parse the test sample by traversal rules and the cluster your test sample ends up in decides the neighbours closest to your test sample.

2. **Merge Sort (a version of it):**
    - If a data sorted in a particular order is inserted in the K-D tree sequentially, then the replication of a K dimensional space is better.
    - The application requires node inserted at each level of the tree to have a median (level % degree'th) value of the elements that are supposed to be its children.
    - This application uses recursive merge sort with a minor modification to achieve the above said precision.

3. **Queues:**
    - One of the requirements while testing is to get the closest of all possible neighbours.
    - To achieve this, the application inserts all the potential elements from the final cluster to the queue in such a way that the nearest element is always at the peek.
    - Thus, after all insertions, when the queue is dequeued, the neighbour that we get are from nearest to farthest.

4. **Graph:**
    - This is another data structure that's used for training purpose in this application, of course, with some minor modifications.
    - There are 2 different training algorithms running simultaneously in this application.
    - Graph is used in supervised learning using my variation of "K mean" technique.
    - During runtime, user can switch the techniques using a dropdown to examine predictions which are outcome of 2 different learning techniques.

Features of application

1.)Training:

- User can provide a path for training data and start training the algorithm.
- Training takes time proportional to the size of training data.
- There are some rules for a directory to qualify as the directory containing training data.
- Rules:
    a. the directory should contain only subfolders.
    b. Name of each subfolder should be a label for training and should signify the data that's present inside the sub-folder. For example, an folder containing images of oranges to train application to recognize orange should be named "Orange".
    c. The feature extraction in this application is implemented with an assumption that the image has either white or transparent background. Thus, the images used to train the algorithm must have either a plain or white background.

2.) Prediction:

- User can browse an image to the application and ask the application to predict what's in the image.
- Application uses the KNN technique on K-D tree to predict what's in the image.
- Again, the application can only predict an image with white or transparent backgrounds.
- User can pick from 2 different methods for prediction. "KD-Tree" & "mean point plot".

3.) Retrieval:

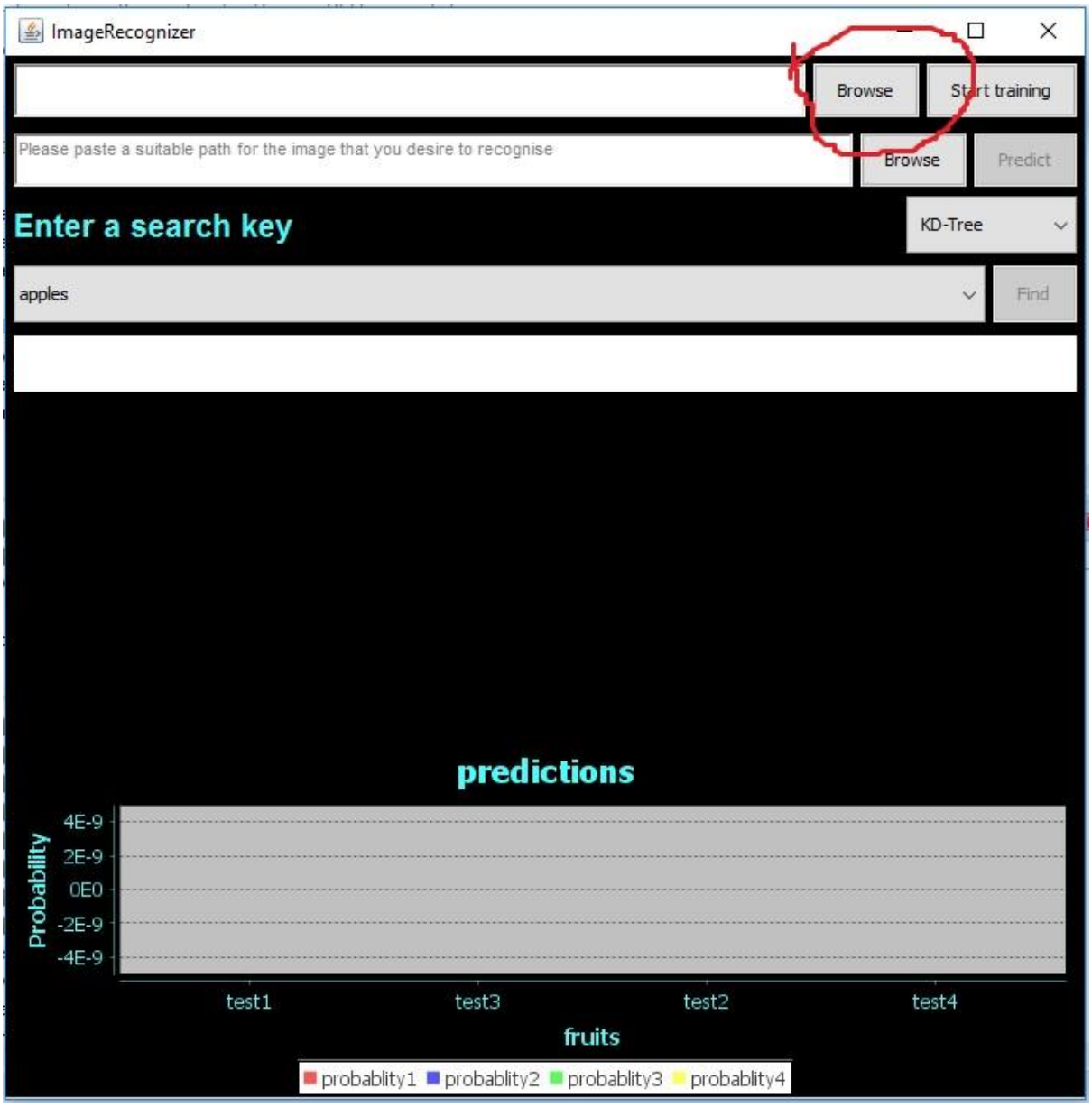
- A dropdown in the application gets filled with the name of the objects that the application has been trained for.
- User can search any random database and retrieve images which can potentially fall under the category selected from the dropdown.

Working and Directions:

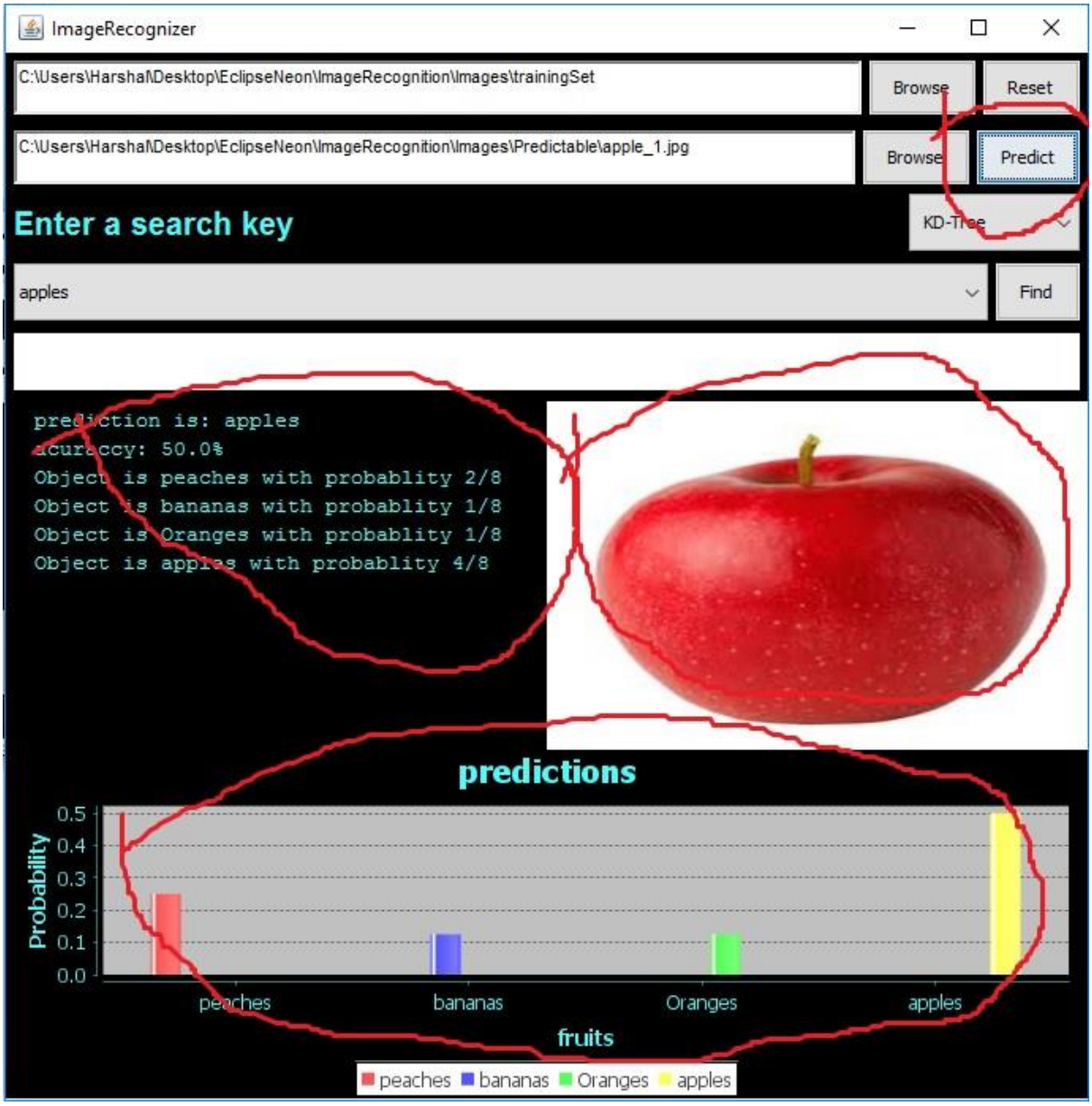
1.) To start with, user needs to train the application. For doing so. User can click the browse button near the topmost label and provide the application with a path to training data.
2.) The training data folder should follow the rules as mentioned in the above section.
3.) After giving the appropriate path, the user can hit the start training button.
4.) When training gets completed, the “start training” button gets converted to “reset” button.
5.) Reset button is used to reset the whole trained data set and train with new samples.
6.) As the training gets completed, the previously disabled “predict” and “Find” buttons get enabled.
7.) Now, user can click the browse button along the 2nd text field from top to provide the application with path for the test image. (application predicts, what object/fruit is present in the image).
8.) Clicking “predict” button would cause the application to make a prediction and display it on the bottom panel. Also, the application shows a graph with variable probabilities for different possible fruits/objects.
9.) Selecting a key from the dropdown and clicking “Find” will cause the application to return a maximum of 15 possible images for the selected key from the database of images whose path is hardcoded.
10.) Modifications can be made in code to provide the database path manually.
11.) Application also displays the precision of the retrieval.
12.) There is a dropdown to select between 2 methods for prediction. Different type of learning goes into both method.

Application:

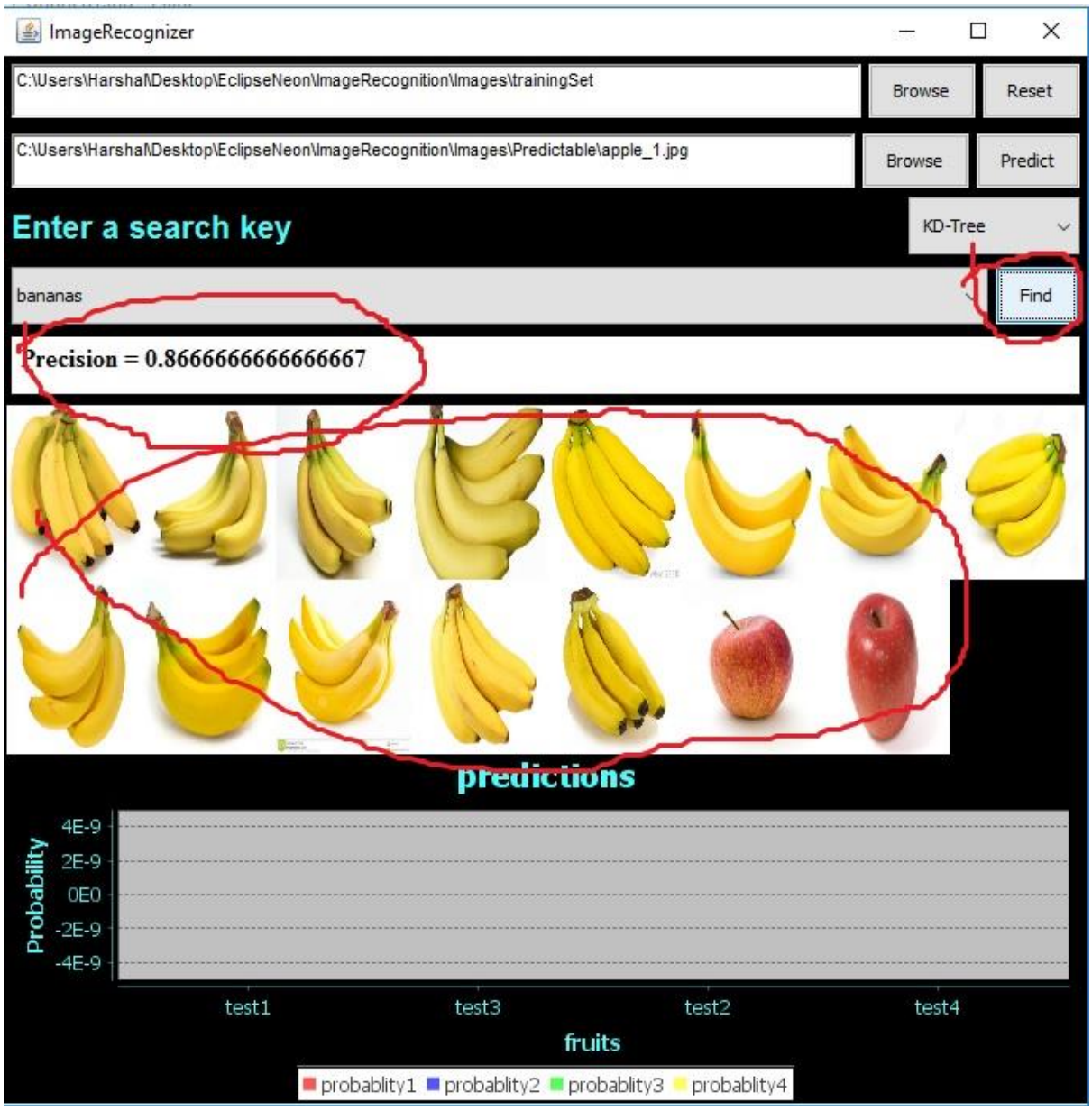
1. As can be seen in the below image, the topmost browse button is used to browse the directory of training data.



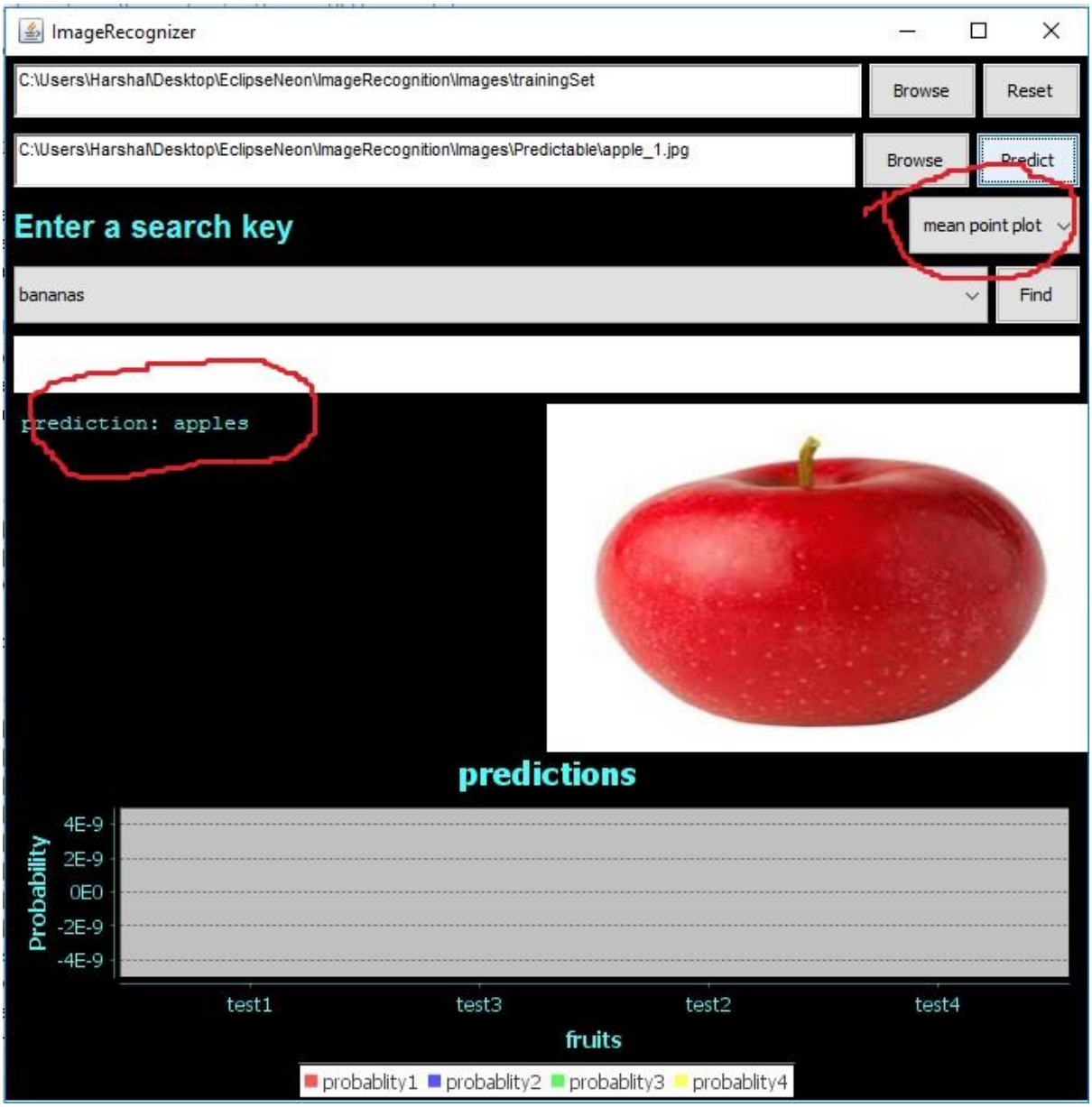
2. After the “start training” button is pressed and the application is trained, the user can use the browse button highlighted below to give application a path to image that user wants the application to detect. The graph in UI shows the probability of image to contain various objects and the image in the bottom right is the one that user wants the application to predict for.



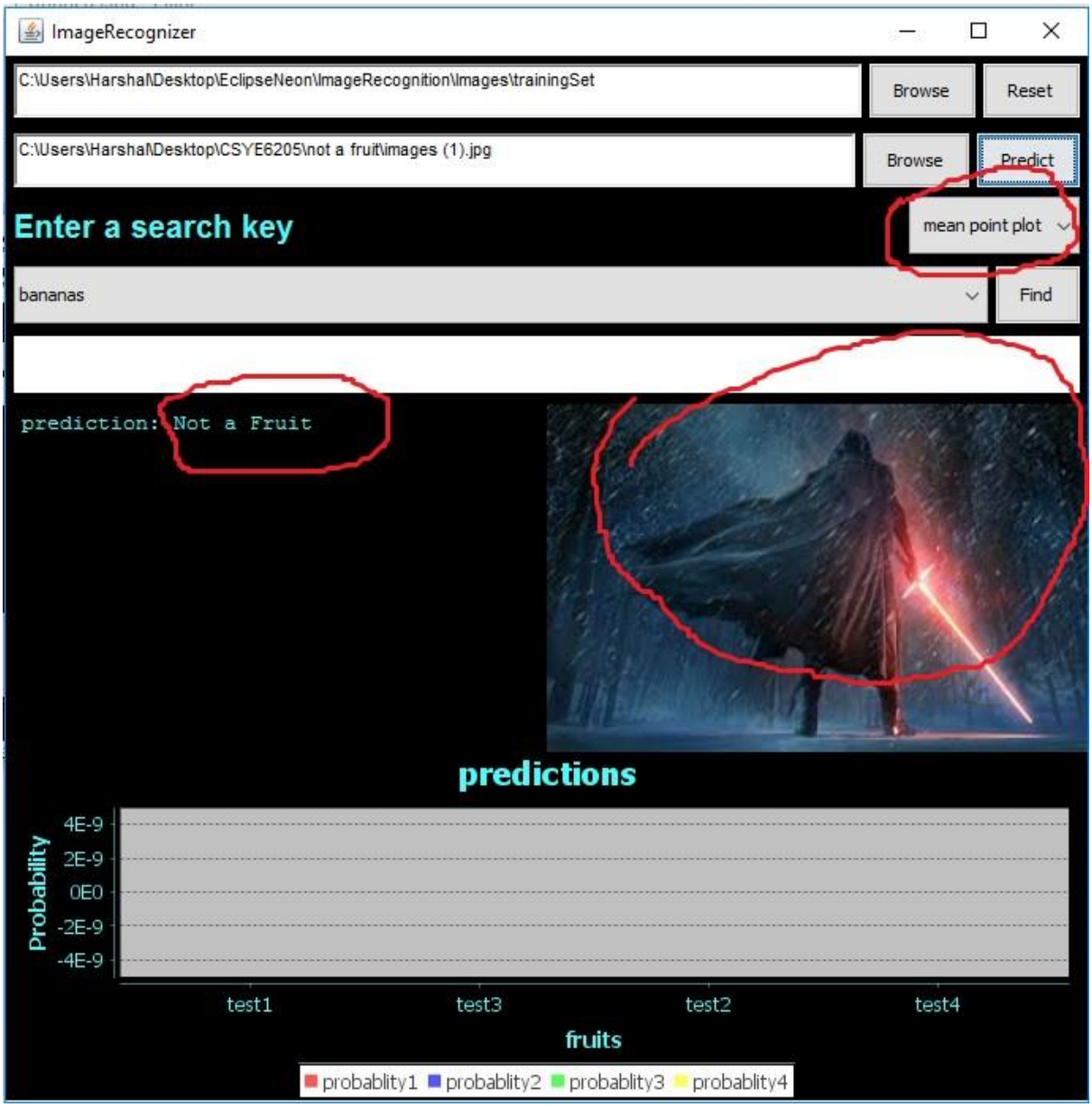
3. The dropdown box highlighted below gets dynamically filled with labels of training set. User can select one and click “Find” to retrieve all the relevant images from any 3rd party data base. The path to 3rd party database is hardcoded in the code and can be changed anytime. Searching Precision is also mentioned in the UI in the highlighted section below.



4. Now, there is one more method for training and prediction. The application has been programmed to get trained by both techniques together. But, while making predictions, user can decide if he wants to make predictions using the “KD-Tree” method or the “Mean point plot” method, which is my version of “K-Mean Method”. Though, there is no probabilistic distribution here, the image either falls in a category or it does not.



5. The advantage of “Mean point plot” is that it can detect if an image is not a fruit, without any negative training set.



Future Scope

1.) Setting a threshold in picking up N Nearest Neighbours can improve the accuracy many folds. But, it requires a larger training set.
2.) Larger training set would lead to bigger clusters in n-dimensional space. This would give better results while making predictions.
3.) Mean of all RGB pixels is a very wage feature when it comes to making comparisons among different objects in an image. A rather distinct feature is more likely to provide better results.
4.) The whole application can be setup on a cloud where with every usage, the algorithm will learn more.
5.) Merging 2 or more learning and predicting techniques can provide interesting results and should be tried.