<Recog – Optical Character Recognition>

CS 293 Final Project Demo

24th November

Anirudh V, 110050055

Vinod Reddy K, 110050060

Outline <for a total of 30 mins>

- Aim of project (1 mins)
- Demo (5 mins)
- Teamwork Details (0.5 min)
- Design Details –Algorithm (5 mins)
- Design Details – Implementation (8 mins)
- Viva (9 mins)
- Transition time to next team (2 mins)

Aim of the project

- An application that takes an image containing characters and recognizes the characters present in them and outputs them into the terminal.

Demo

- *Now explain the program further while doing the demo*
  - *Tip: one member can explain, one member can do demo*

Teamwork Details

- Work Division:
  - Vinod Reddy K: *Image Manipulation, basic OpenCV functions regarding image processing, linking the training and recognizing parts of the program, overall class and code structure, Coding the whole Image processing part.*
  - Anirudh V: *OpenCV data structures, Basic OpenCV functions regarding Machine Learning, classifiers, Coding the whole training part, Database development, Designing the Box class for perfect ordering,*

Overall Contribution by Vinod Reddy K	Overall Contribution by Anirudh V
50%	50%

Design Details

- Algorithm
  - **Detecting Characters**: In the first stage, we have to detect the characters in the image given and mark them.
  - **Training Characters**: After detecting the characters and marking them, we have to train the program to recognize these characters.
  - **Recognizing Characters**: After training, the user provides the test image whose characters are detected and recognized.

Design Details

- Algorithm
  - **Ordering of Characters:** When we detect the contours we have used an algorithm which detects from top to bottom. When it encounters a black pixel, it creates a ‘blob’ and joins all the adjoining black pixels to the ‘blob’ and detects a contour. Due to this, we detect the tallest character in the line first and then the shorter ones. Thus, we had to prevent this and detect the characters in the order that they were present in.

Design Details

- **Implementation**
  - We have used OpenCV (Open Computer Vision library) for the detection and training part. We have used Machine Learning (K-Nearest Classifier) for training the characters. More on K-Nearest Classifier in following slides.
  - We have used data structures such as vector, CvMat(matrix data structure in OpenCV), CvSeq and many others for data storage and manipulation. We have used priority queue for a very important operation which will be described in the following slides.

Algorithm Design

- **Detection of Characters:** We have to detect the characters present in the image and mark them for future operations.
- We threshold the image to turn it into a Binary Image so that we don’t have to deal with any blurredness/noise. Then we find contours present in the image and draw bounding rectangles around them. These contours represent the boundaries of the characters.

Algorithm Design

- **Detection of Characters:** We have to detect the characters present in the image and mark them for future operations.
- Whenever we find a contour, we identify both the inner and outer boundary as contours. But we need only the outer contour, so we disregard the inner contours by a predicate defined in OpenCV (CV_IS_SEQ_HOLE(contour))

Algorithm Design

- **Detection of Characters:** We have to detect the characters present in the image and mark them for future operations.

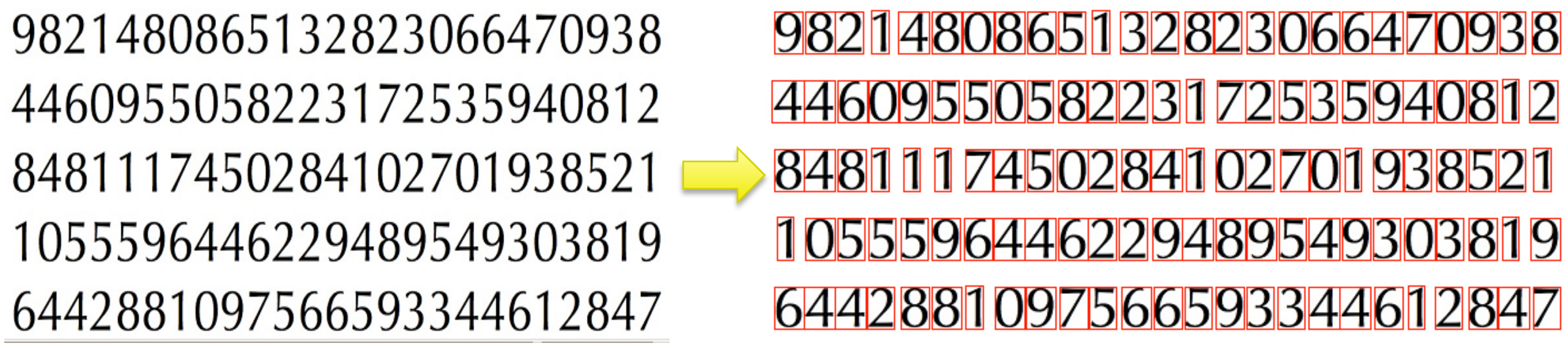


Image Source: StackOverFlow.com

Algorithm Design

- **Ordering of Characters:** We have to detect the characters present in the image in the order that they are present.
- We defined a priority queue and defined a different class for the bounding rectangle of character with the operator< overloaded so that when we push the object into the priority queue they get ordered according to the position of the rectangle in the image thus resulting in getting an ordered set of characters.

Algorithm Design

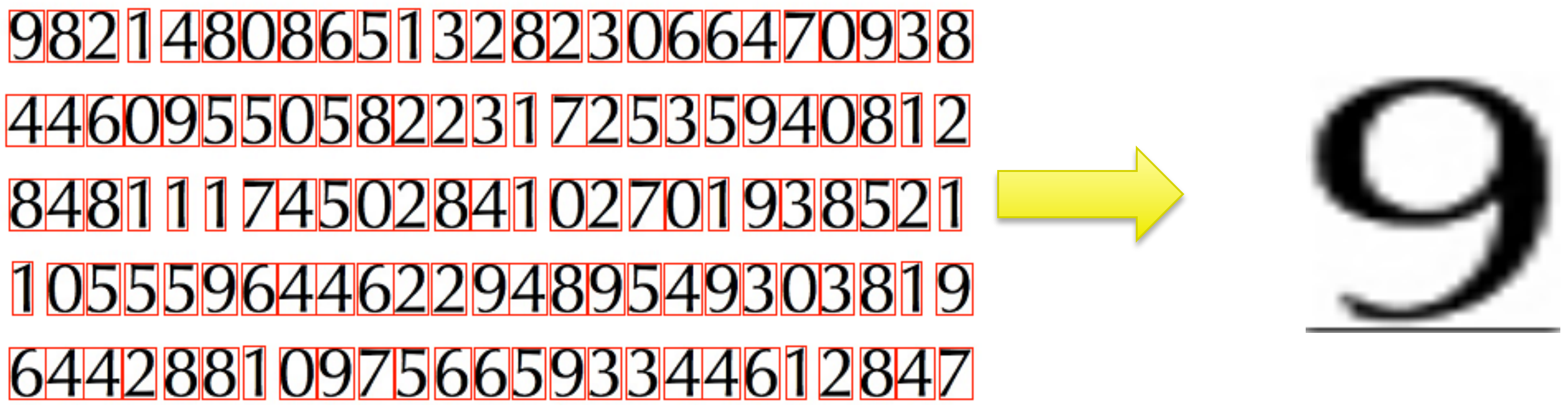
- **Training of Characters:** After detecting and marking the characters of the training image, we train the program to recognize these characters
- We isolate the characters in the training image and resize each of them into a 20x20 image. We store the 400 pixels of this character image in a vector. We read the corresponding label for this character from a file and write the vector and label into separate files for future use.

Algorithm Design

- **Training of Characters:** After detecting and marking the characters of the training image, we train the program to recognize these characters
- We read vectors and labels from the files and store them in matrices(CvMat) and load them into a K-Nearest Classifier object. This object will act as a classifier from now on.

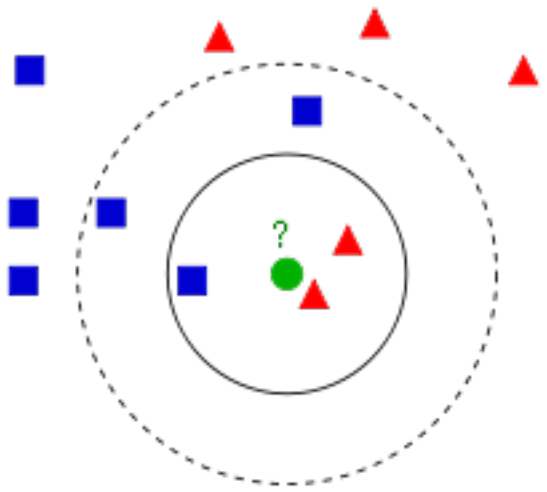
Algorithm Design

- **Training of Characters:** After detecting and marking the characters of the training image, we train the program to recognize these characters



Algorithm Design

- **K-Nearest Neighbor Algorithm:** We classify based on closest training examples in the feature space. For a specific value of K, we observe the K nearest neighbors in the feature space and output the label which is present in majority among the neighbors.



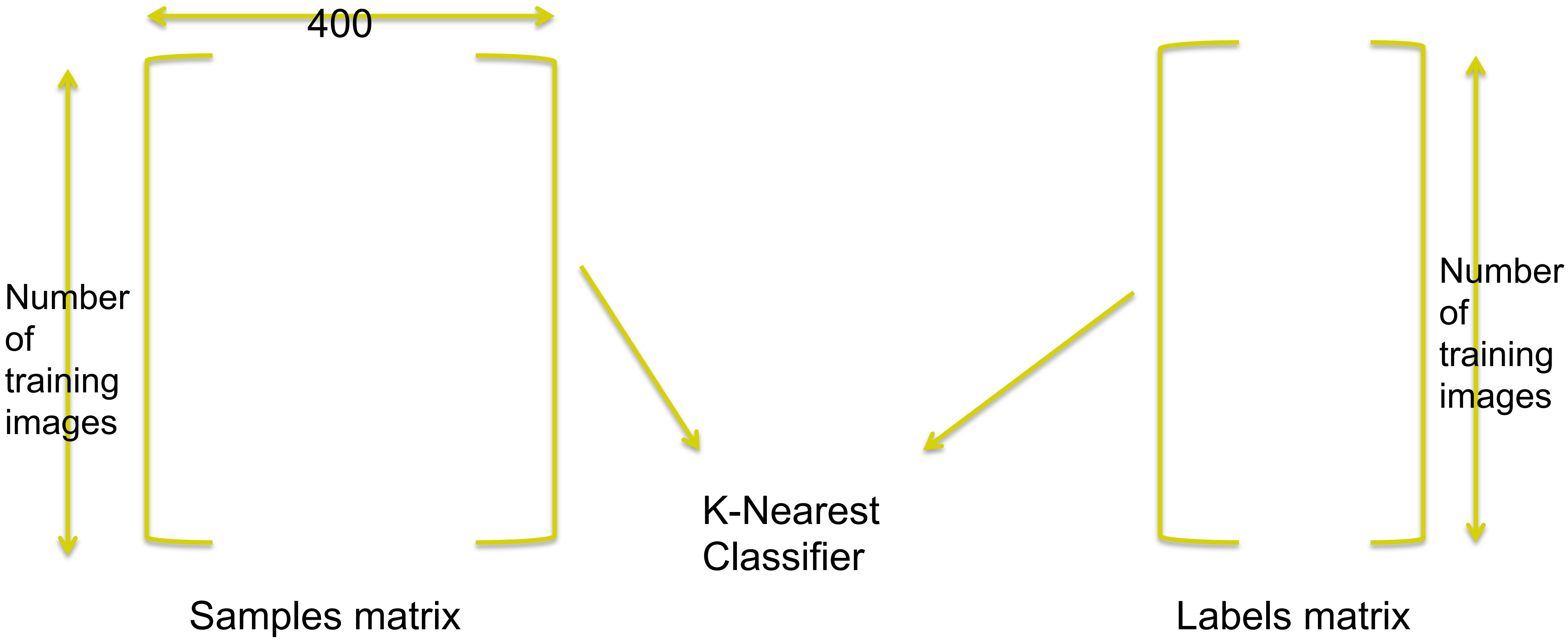
Source: Wikipedia

Algorithm Design

- **Recognizing Characters:** When the user presents the test image, our program detects and recognizes the characters in the image.
- After, detecting the characters in the test image and marking them, we pass the 400 element feature vector of each 20x20 character image to the K-Nearest Classifier which outputs the label of the character according to the K-Nearest Neighbor algorithm described in the previous slide.

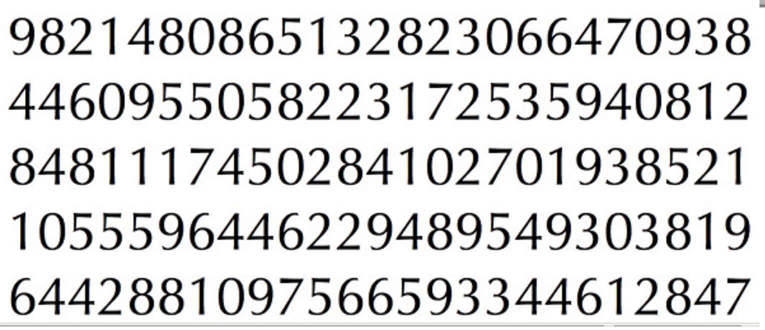
Algorithm Design

- **Recognizing Characters:** When the user presents the test image, our program detects and recognizes the characters in the image.

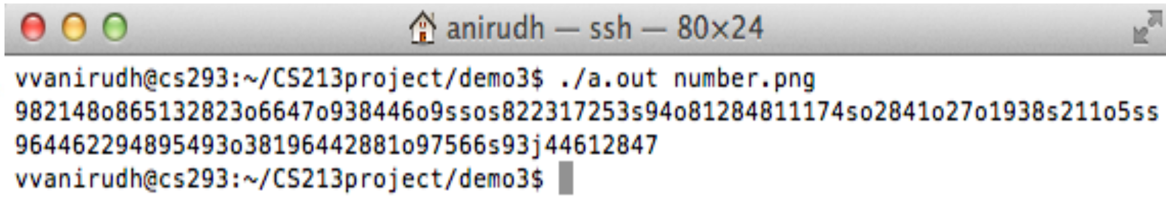


Algorithm Design

- **Recognizing Characters:** When the user presents the test image, our program detects and recognizes the characters in the image.

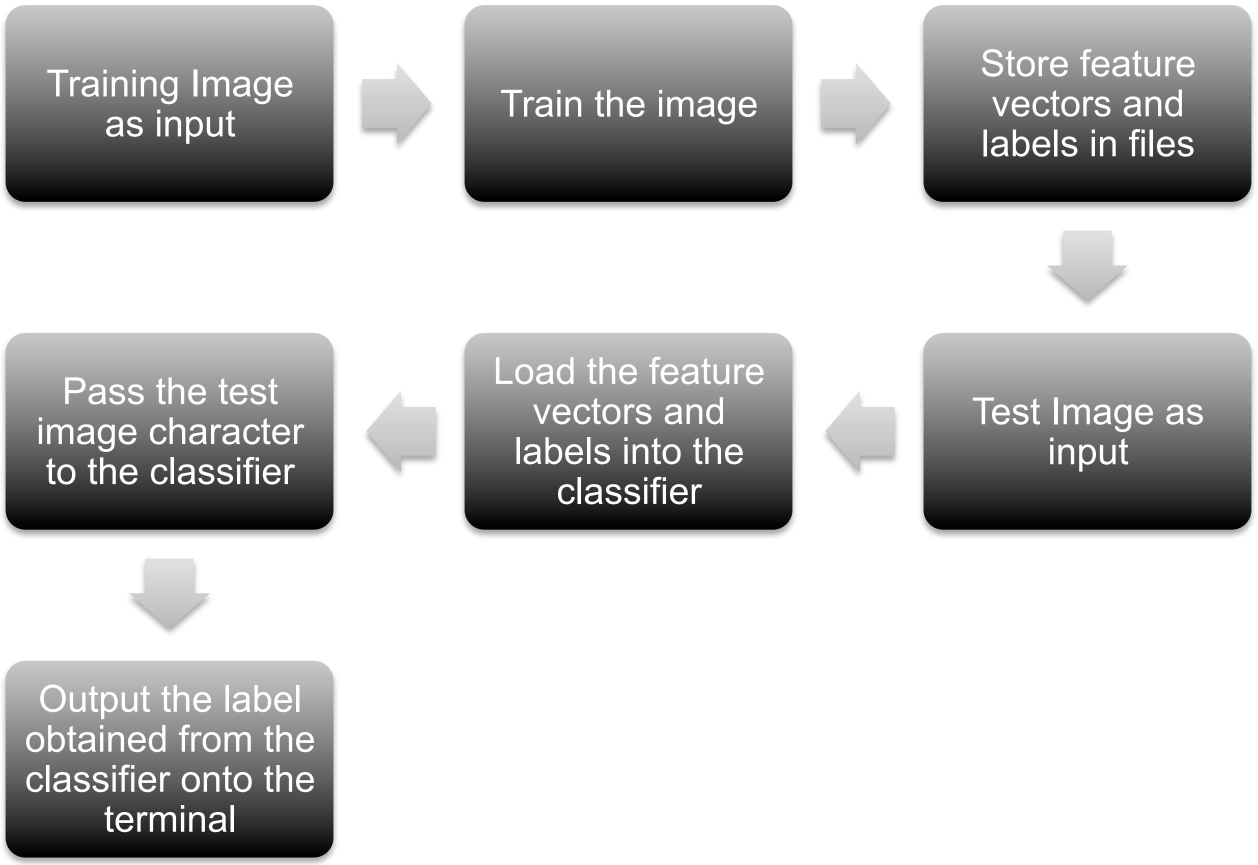


```
9821480865132823066470938
4460955058223172535940812
8481117450284102701938521
1055596446229489549303819
6442881097566593344612847
```



```
vvanirudh@cs293:~/CS213project/demo3$ ./a.out number.png
982148o865132823o6647o938446o9ssos822317253s94o81284811174so2841o27o1938s211o5ss
964462294895493o38196442881o97566s93j44612847
vvanirudh@cs293:~/CS213project/demo3$ 
```

Project Workflow

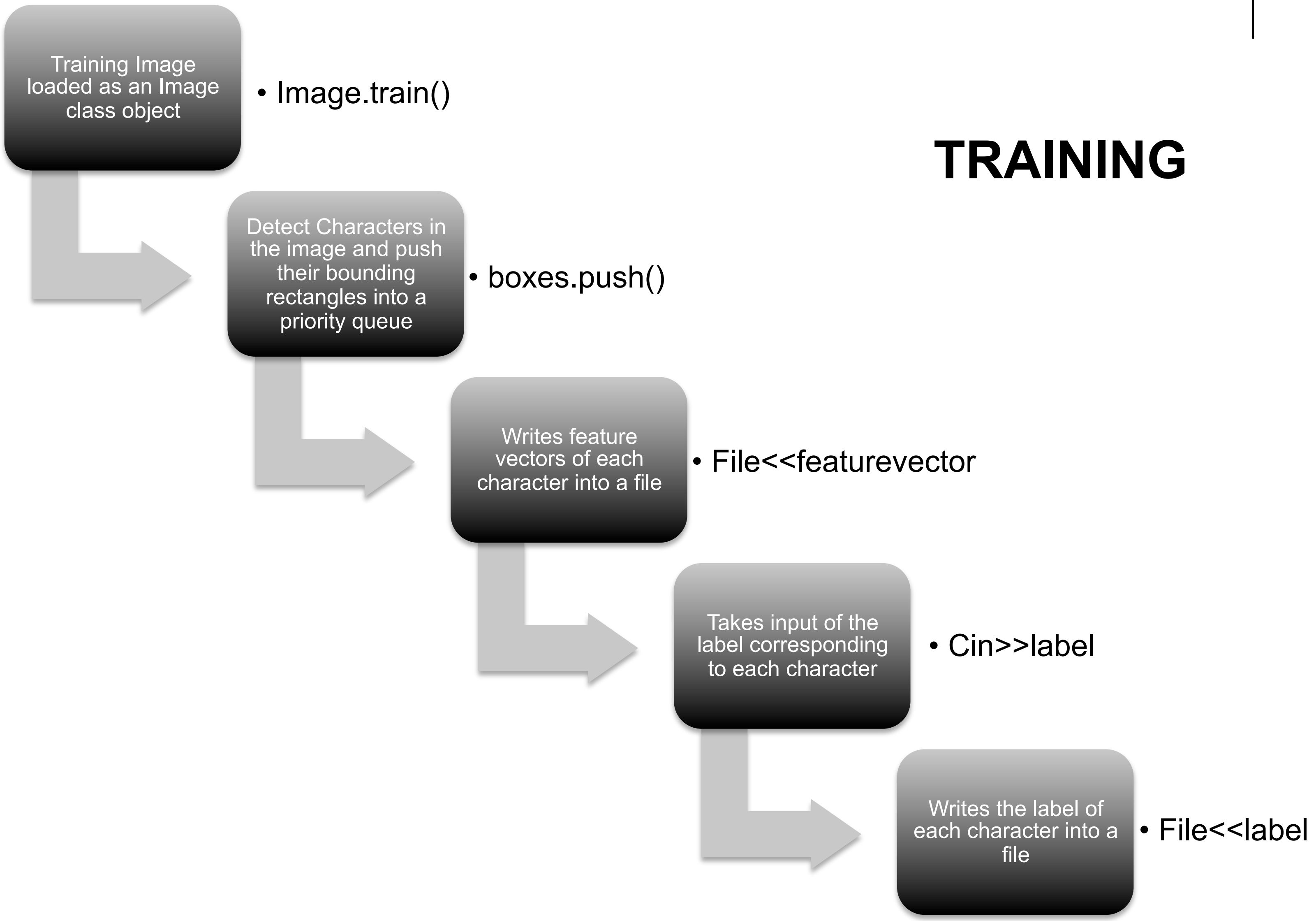

Training Image as input
Train the image
Store feature vectors and labels in files
Test Image as input
Load the feature vectors and labels into the classifier
Pass the test image character to the classifier
Output the label obtained from the classifier onto the terminal

Class Design – High level

- For simplicity in understanding and ease of collaborating, we have divided the whole project into two parts: Image Processing, Machine Learning.
- The class “Image” deals with all the image processing functions used. Any manipulation done in the image is done within the Image class.
- The class “TrainedData” deals with all the training and recognizing part of the project. Any function regarding classifiers, data storage and training is done within the TrainedData class.
- As you can observe, we have divided the work between us in the same way making our collaboration simpler.

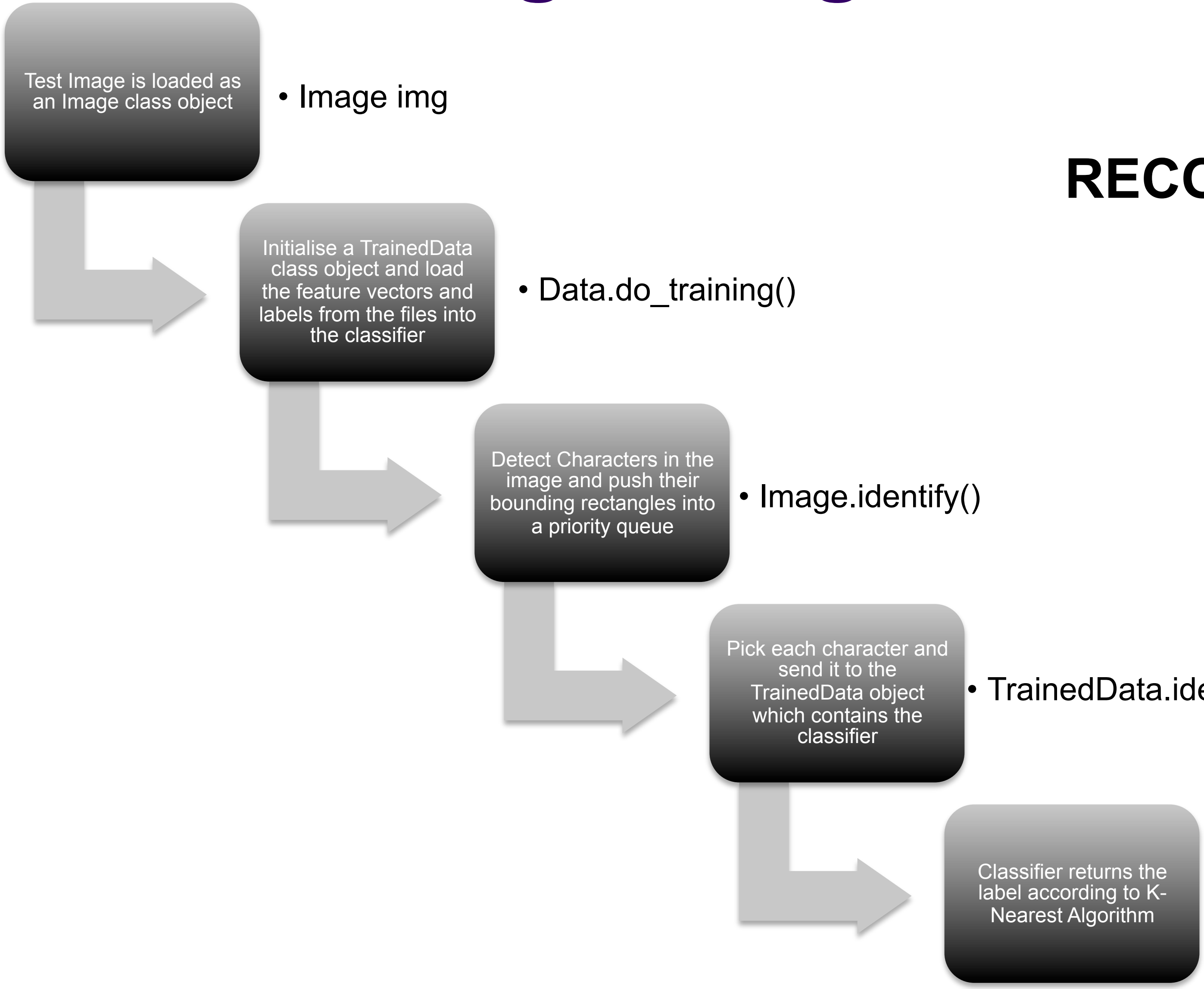
Class Design – High level

TRAINING


Training Image loaded as an Image class object
• Image.train()
Detect Characters in the image and push their bounding rectangles into a priority queue
• boxes.push()
Writes feature vectors of each character into a file
• File<<featurevector
Takes input of the label corresponding to each character
• Cin>>label
Writes the label of each character into a file
• File<<label

Class Design – High level

RECOGNIZING



Test Image is loaded as an Image class object

- Image img

Initialise a TrainedData class object and load the feature vectors and labels from the files into the classifier

- Data.do_training()

Detect Characters in the image and push their bounding rectangles into a priority queue

- Image.identify()

Pick each character and send it to the TrainedData object which contains the classifier

- TrainedData.identify(character)

Classifier returns the label according to K-Nearest Algorithm

- Classifier.find_nearest(character)

Class Design - Details

Image:

- Description: Contains the information and attributes of an image that is loaded into the program. This class has all the methods that we use to detect, recognize and train images. This is the class in which the image processing and manipulation part is done.
- Data Members:
  - ➢ IplImage : The OpenCV image data structure. This is the data member in which the image is stored.
  - ➢ Priority Queue: The priority queue that stores the bounding rectangles of the contours.

Class Design - Details

- CvSeq* : The sequence of contours present in the image.
- uchar* : The data structure in which the data regarding the image is present(i.e. the pixel values)

- Methods:
  - box() : Draws the bounding rectangles on the contours found in the image.
  - display() : displays the image in a separate window.
  - train() : Trains the image.
  - identify(TrainedData& data) : identifies the characters in the image using ‘data’.

Class Design - Details

TrainedData:

- Description: Contains the information about the trained data. This class is the place where all the characters that have been trained upon are stored in their feature vector form.
- Data Members:
  - ➢ CvMat* samples : The OpenCV matrix data structure. This is the data member in which all the trained images feature vectors are stored.
  - ➢ CvMat* labels: The OpenCV matrix data structure in which all the corresponding labels of the trained images are stored in.

Class Design - Details

- CvKNearest : The K-Nearest classifier in which the data is loaded.
- CvMat* identifysample: The data structure in which the feature vector of the test image is present(i.e. the pixel values)
- CvMat* identifylabel: The data structure in which we obtain the label of the test image.

- Methods:
  - identify(vector<int> sample) : This method identifies the given feature vector and returns the corresponding label.
  - do_training() : This method does the training on the data present in samples and labels.

Class Design - Details

Box:

- Description: The class containing the bounding rectangle of a character. This class is defined to ensure ordering of characters.
- Data Members:
  - ➢ CvRect : OpenCV data structure to store a rectangle. This stores the bounding rectangle of a character.
- Methods:
  - ➢ setRect(CvRect) : sets the rectangle to the given rectangle.
  - ➢ getRect(): returns the rectangle present in the class.

Class Design - Details

- operator< : Overloads the < operator to ensure that when these ‘boxes’ are pushed into a priority queue, they get ordered according to their position in the original image.

Data Structures Used

Purpose for which data structure is used	Data Structure Used	Whether Own Implementation or STL
Perfect ordering of contours	*Priority queue*	*STL*
Storing of feature vector and labels	*CvMat*	*OpenCV*
Storing of contours	*CvSeq*	*OpenCV*
Storing of feature vector of a single image	*vector*	*STL*
Storing bounding rectangle	*Box*	*Own*

Source Code Information

File Name	Brief Description	Author (Team Member)
Image.h	*Declares Image class and methods*	*Vinod Reddy K*
Image.cpp	*Implements Image class methods*	*Vinod Reddy K*
TrainedData.h	*Declares TrainedData class and its methods*	*Anirudh V*
TrainedData.cpp	*Implements TrainedData class methods*	*Anirudh V*
Box.h	*Declares Box class and its methods*	*Anirudh V*

Source Code Information

File Name	Brief Description	Author (Team Member)
Box.cpp	*Implements Box class methods*	*Anirudh V*
train.cpp	*Main function which trains on a set of images*	*Anirudh V*
recog.cpp	*Main function which recognizes the character in a test image*	*Vinod Reddy K*

Brief Conclusion

- Although this OCR performs well on good quality images, it fails on noisy images where two/more characters may be combined. Thus we have demonstrated the project only on good quality images.

Thank You – Questions?

< time for me to do viva>