Kritika Srivastava kxs161730 Satwant Singh sxs149531

**Handwritten Grocery List Detection**

[Comparison between SVM and KNN]

**INTRODUCTION**

In the running world, there is growing demand for the software systems to recognize characters in computer system when information is scanned through paper documents as we know that we have number of newspapers and books which are in printed format related to different subjects. These days there is a huge demand in “storing the information available in these paper documents in to a computer storage disk and then later reusing this information by searching process”. One simple way to store information in these paper documents in to computer system is to first scan the documents and then store them as IMAGES. But to reuse this information it is very difficult to read the individual contents and searching the contents form these documents line-by-line and word-by-word. The reason for this difficulty is the font characteristics of the characters in paper documents are different to font of the characters in computer system. As a result, computer is unable to recognize the characters while reading them. This concept of storing the contents of paper documents in computer storage place and then reading and searching the content is called DOCUMENT PROCESSING. Sometimes in this document processing we need to process the information that is related to languages other than the English in the world. For this document processing we need a software system called **CHARACTER RECOGNITION SYSTEM.**

**OVERVIEW**

The objective of this project is to design and implement an order generation application that detects a hand-written grocery shopping list and converts it to a digital item list using linear **SVM** algorithm coupled with **HOG** feature extraction. The data set contains different 62 different characters (English alphabets capital and small, Digits 0-9) written in 55 different styles. In particular, we will apply the tools of machine learning to detect the handwritten characters. We will use segmentation, HOG feature extraction and linear SVM model to obtain better performance from any of the constituent machine learning algorithms.

The input here is the images of characters written in 55 different styles. The final output, the digitalized form of the recognized characters, is obtained by extracting its HOG features and detecting the characters by matching the HOG features from the trained linear SVM model.

Tools Used: OpenCV2 ,Python 3.0

Libraries: Scikit-learn for HOG, SVM, KNN,numpy

Dataset: The Chars74K  <http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/EnglishHnd.tgz>

**Algorithms**

### **Histogram of Oriented Gradients**

It is a [feature descriptor](https://en.wikipedia.org/wiki/Feature_descriptor) used in [computer vision](https://en.wikipedia.org/wiki/Computer_vision) and [image processing](https://en.wikipedia.org/wiki/Image_processing) for the purpose of [object detection](https://en.wikipedia.org/wiki/Object_detection). The essential thought behind the histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing. The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation.

### **K Nearest Neighbors Algorithm**

K Nearest Neighbors algorithm is an **Instance based learning** algorithm where a training sample is classified based on its nearest neighbor. It is a type of **lazy learning algorithm**. Thus when a query is made to such a system, the prediction about the class is provided by finding similarity of the test case with the training dataset rows. Such systems are good for datasets which have noise and contain lots of features. Thus the prediction is done by the majority of the vote of its neighbor where k is the number of nearest neighbors to take under consideration.

This implementation of the KNN Algorithm uses the **Euclidean Distance** for calculating the distance between the instances. Here we have used the value of k as 3. We witnessed that the accuracy was 70.37% for the first image and 56.6% for the 2nd image.

### **SVM Algorithm**

Support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. We stumbled upon the idea of using SVM after reading a paper by Joana Raquel Cerqueira Da Silva [4] where she used SVM-MC to obtain accuracies as high as 96%. Support vector machines gave us an whopping accuracy of **85.18%** for the 1st image and **80%** for the 2nd image.

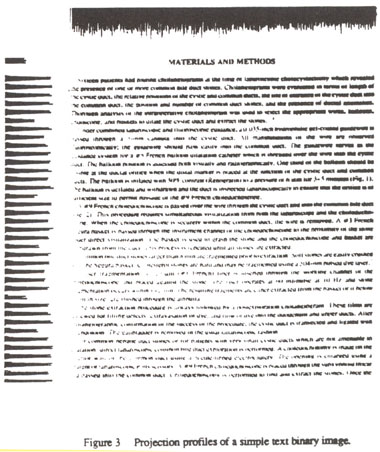
**Processing Training Data**

The training data consists of images of all characters from A-Z, a-z and 0-9. These images are trained using HOG features. Each image is first binarized and then its contours are detected and then a HOG matrix of each image is produced. All the images HOG values and their corresponding labels are passed to the SVM or KNN classifier to fit.

**Processing Input Data**

### **Segmentation with HOG feature extraction**

The input image is segmented into several images based on the number of lines of text in the image. For this we first binarize the image using thresholding and then use a horizontal histogram. It is like a histogram itself, but shows where there are lines and where there are blank spaces. Now we divide the image at blank lines and we get each line as a different image. Below is an example of horizontal histogram.

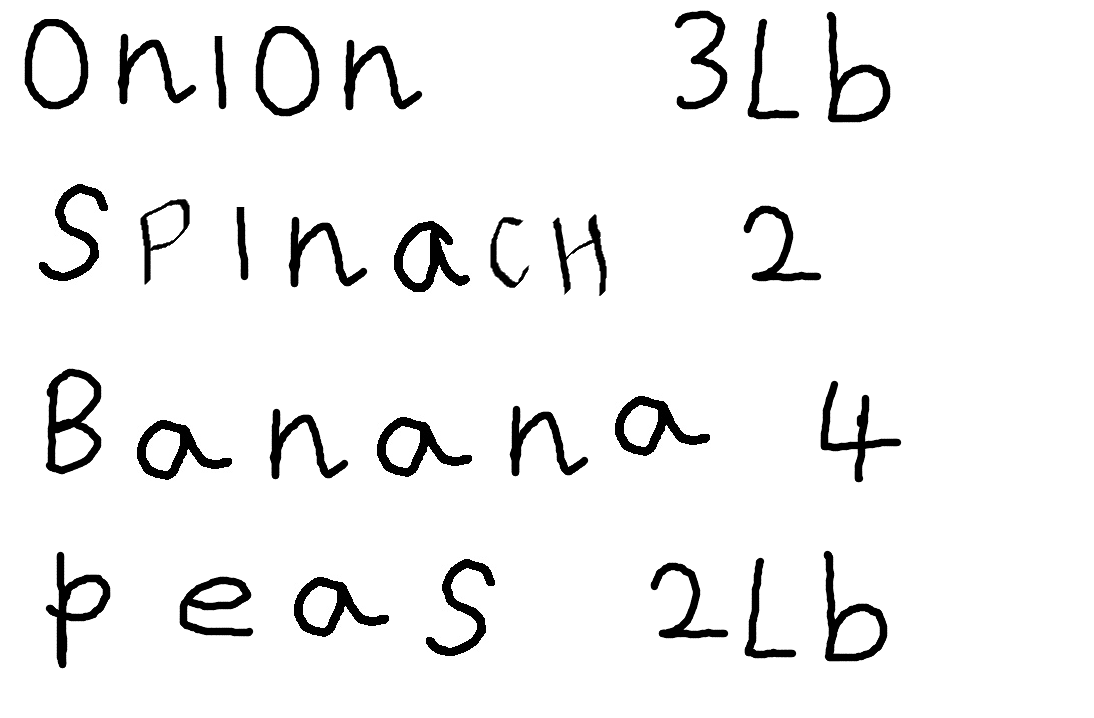


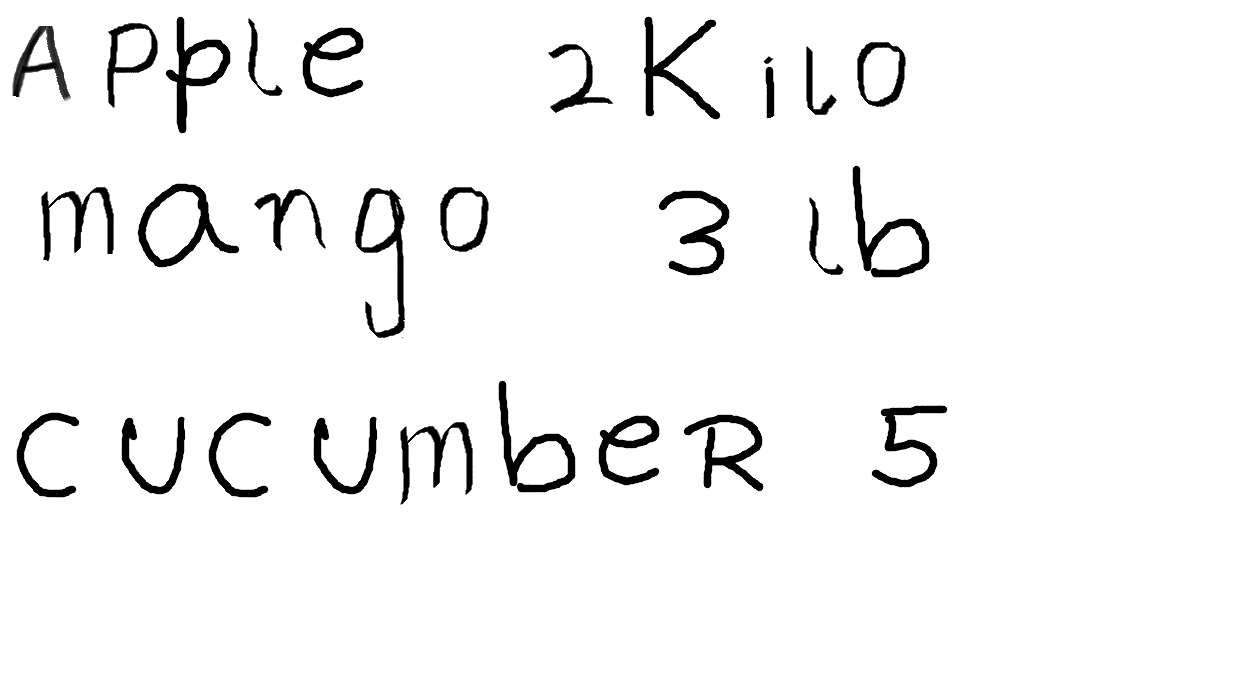
Now for each line, find horizontal histogram. Before that, try to do some dilatation and erosion, so that all letters are grouped together. Then you can find connected components on each line to get each word. Then draw boundaries.

Below image shows both horizontal and vertical histograms:



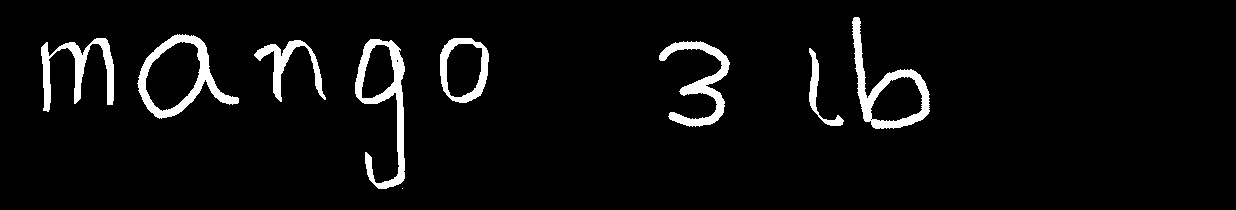
### **Input Images**

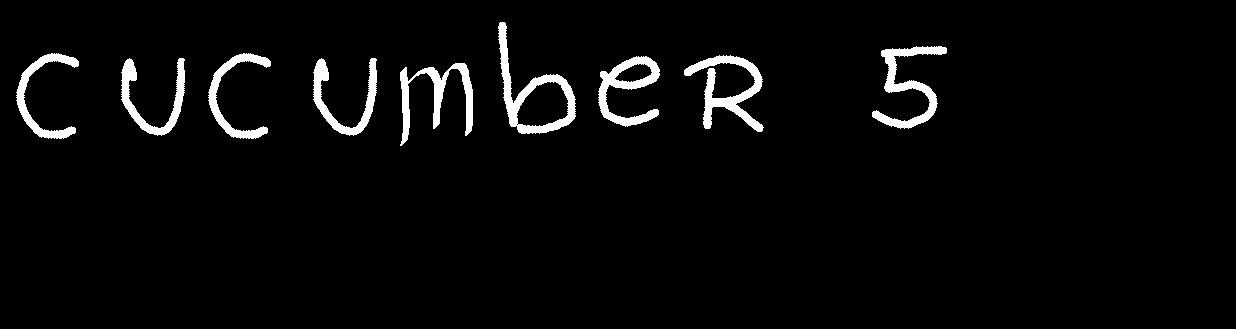


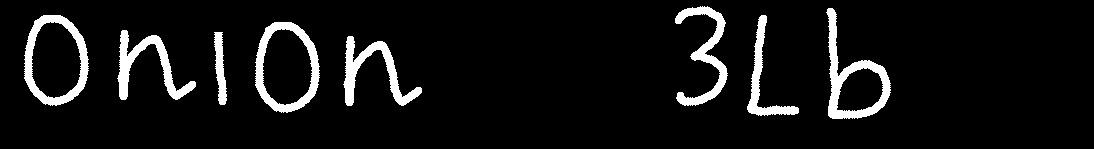
****

### **Output after Segmentation**

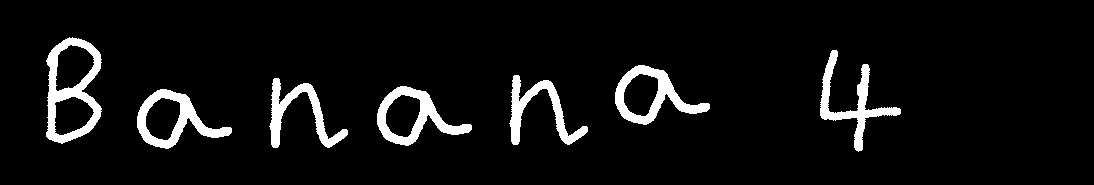








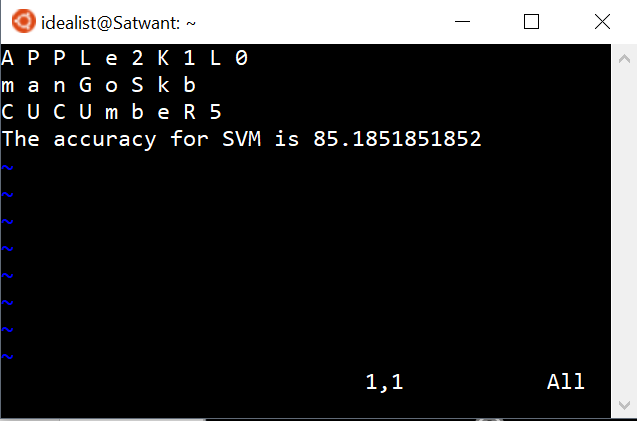
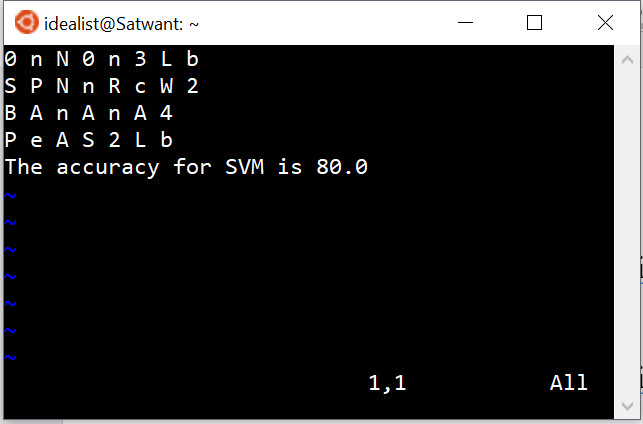




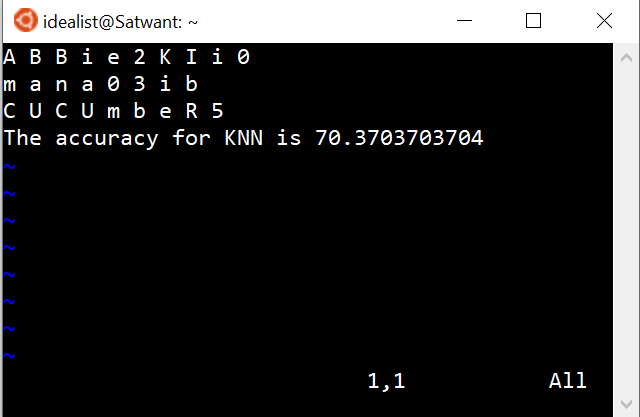
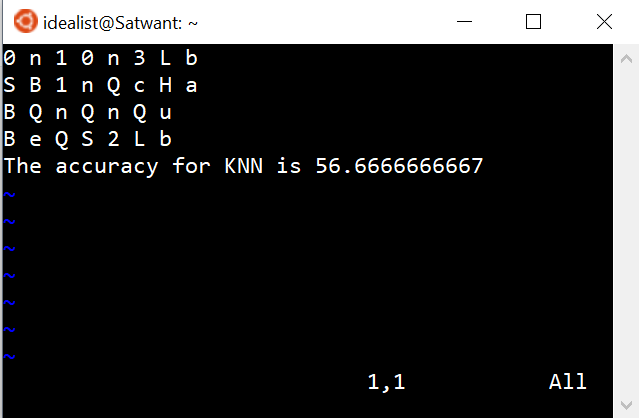


### **Final Output**

Text Prediction using SVM Classifier:



Text Prediction using KNN Classifier:



**Conclusion**

We experimented the recognition with SVM and KNN classifiers. Accuracy of SVM was much higher than KNN.

Also, KNN trained set required huge amount of space to store them as it stores all data points. On the other hand, the trained model by SVM Classifier occupied much lesser space.

**References**

<https://www.cs.rochester.edu/~brown/242/assts/termprojs/vision.pdf>

<http://www.wseas.us/e-library/conferences/2009/cambridge/ISPRA/ISPRA45.pdf>

<https://en.wikipedia.org/wiki/Image_segmentation>

<http://francescopochetti.com/text-recognition-natural-scenes/>

<http://www.sersc.org/journals/IJSIP/vol9_no1/5.pdf>

<http://www.wseas.org/multimedia/journals/signal/2015/a045714-403.pdf>

<http://www.danvk.org/2015/01/07/finding-blocks-of-text-in-an-image-using-python-opencv-and-numpy.html>

<http://www.sersc.org/journals/IJSIP/vol8_no4/14.pdf>

<http://www.ijteee.org/final-print/nov2013/Text-Detection-From-Documented-Image-Using-Image-Segmentation.pdf>

<https://pdfs.semanticscholar.org/5b6b/e3357dbdbac38e92515a7b7aebb7e622f635.pdf>

<http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_svm/py_svm_opencv/py_svm_opencv.html>