

CHARACTER RECOGNITION SYSTEM

The above mini project is submitted in partial fulfillment of
the requirements of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

Prepared by

**ANKITA BHUTRA
MILAN CHANDIRAMANI
PAVAN AGARWAL**

Guide

**Mrs. SHILPA VERMA
(Associate Professor, Department of
Computer Engineering, TSEC)**



**Computer Engineering Department
THADOMAL SHAHANI ENGINEERING
COLLEGE
UNIVERSITY OF MUMBAI
2019-20**

CERTIFICATE

This is to certify that the mini project titled
“Character Recognition System” is a bonafide
work of

Roll Number	Semester	Name
04	VI	Pavan Agarwal
15	VI	Ankita Bhutra
19	VI	Milan Chandiramani

(Name and Sign)

Guide

Internal Examiner

External Examiner

TABLE OF CONTENTS

List Of Illustrations	iv
Chapter 1 Introduction	01
1.1. General Introduction	01
1.2. Problem Definition	01
1.3. Domain	02
1.4. Need	02
1.5. Scope	03
1.6. Application	03
Chapter 2 Design	04
2.1 Literature Survey	04
2.2 Techniques Used	06
2.3 Requirements	09
2.4 Design	12
Chapter 3 Implementation	14
3.1 Algorithm Used	14
3.2 Output	18
3.3 Result	19
Chapter 4 Conclusion and Future Scope	20
Chapter 5 References	22
Chapter 6 Acknowledgement	24

LIST OF ILLUSTRATIONS

Fig. 2.1.	Dimensionality Reduction using PCA	07
Fig. 2.2.	Variance of Principal Components	08
Fig. 2.3.	Information interpreted by PCA	08
Fig. 2.4.	Visual Breakdown of EMNIST Dataset	10
Fig. 2.5.	Block diagram of Software Design	12
Fig. 3.1.	Algorithm Functioning	14
Fig. 3.2.	Character Prediction 1	18
Fig. 3.3.	Character Prediction 2	18
Fig. 3.4.	Accuracy on Character Recognition	19
Fig. 4.1.	Classes similar to 'U'	20

1.INTRODUCTION

1.1 General Introduction

Character recognition is a field of research in artificial intelligence, computer vision, and pattern recognition. A computer performing handwriting recognition is said to be able to acquire and detect characters in paper documents, pictures, touch-screen devices and other sources and convert them into machine-encoded form. Its application is found in character recognition, transcription of handwritten documents into digital documents and more advanced intelligent character recognition systems. Character recognition can be thought of as a subset of the image recognition problem.

This application is useful for recognizing all characters and digits of the English language given as in the input image. Once input image of character is given to the proposed system, then it will recognize input character which is given in image. Recognition and classification of characters are done by k-nearest neighbours and support vector machines. The main aim of this project is to effectively recognize a particular character of type format using the SVM-KNN approach. Basically, the algorithm takes an image (image of a character) as an input and outputs the likelihood that the image belongs to different classes (the machine-encoded characters and digits).

1.2 Problem Definition

The problem with this project is to classify handwritten characters. The goal is to take an image of a handwritten character and determine what that character is. The characters consist of small and capital English alphabet set a,...,z; A,...,Z and the digits 0,...,9.

The Support Vector Machines (SVMs) and Nearest Neighbor (NN) techniques are combined to solve the problem. The tasks involved are the following:

1. Preprocess the EMNIST balanced dataset
2. Train a classifier that can categorize the handwritten digits
3. Apply the model on the test set and report its accuracy

The dataset for this problem is downloaded from [kaggle](#), which was taken from the famous EMNIST (an Extension of Modified National Institute of Standards and Technology) dataset.

1.3 Domain

The problem definition of Character Recognition belongs to the domain of **Image processing through Machine Learning**. It lies in the field of artificial intelligence, computer vision, and pattern recognition. Image processing is an absolutely crucial area of work for those groups and industries that are working in areas such as medical diagnostics, astronomy, geophysics, environmental science, data analysis in laboratories, industrial inspection, etc. Although not originally developed for users who are visually impaired, Character Recognition technology has become an aid for inputting documents quickly by and for users with vision impairments. A complete Character Recognition system consists of a scanner, the recognition component, and a software that interacts with the other components to store the computerized document in the computer.

1.4 Need

The need of Character Recognition is to simulate the human reading capabilities so that the computer can read, understand, edit and work as humans do with text. Character Recognition has been one of the most fascinating and challenging research areas in the field of image processing and pattern recognition in recent years . It contributes immensely to the advancement of automation process and improves the interface between man and machine in numerous applications.

Also, it reduces and/or eliminates costly data entry by automatically grabbing information you need from paper and putting it where it needs to go. Enabling entirely new ways to process documents that can eliminate “human touches”, thereby reducing costs and dramatically reducing processing times.

1.5 Scope

- System will be designed in a way to ensure offline Handwritten Recognition of English characters.
- Old handwritten literature can be restored in digital form.
- Use of SVM - KNN for classification .
- Large number of training data sets will improve the efficiency of the suggested approach.

1.6 Application

Application of Handwritten Character Recognition are as given below:

- Digital Character Conversion
- Meaning Translation
- Content Based Image Retrieval
- Keyboard Spotting
- Signboard Translation
- Text-to-Speech Conversion.

2. DESIGN

2.1. Literature Survey

2.1.1. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition

In this paper, the authors have proposed a hybrid of the two methods viz., k- nearest neighbour(KNN) algorithm and support vector machines(SVM) which deals naturally with the multiclass setting, has reasonable computational complexity both in training and at run time, and yields excellent results in practice. The basic idea is to find close neighbors to a query sample and train a local support vector machine that preserves the distance function on the collection of neighbors. The authors have applied their method to benchmark data sets for shape and texture classification (MNIST, USPS, CURET) and object recognition (Caltech- 101). [11]

2.1.2. CHARACTER RECOGNITION IN NATURAL IMAGES

In this paper, the authors have focused on recognizing characters in situations that would traditionally not be handled well by OCR techniques. They have created an annotated database of images containing English and Kannada characters and have applied nearest neighbour and SVM classification algorithms to assess the performance of various features on their database. The authors have demonstrated that the performance of the proposed method, using as few as 15 training images, can be far superior to that of commercial OCR systems. [12]

2.1.3. Shape Matching and Object Recognition using Shape Contexts

In this paper, the authors have presented a novel approach to measure similarity between shapes and exploit it for object recognition. Given the point correspondences, they have estimated the transformation that best aligns the two shapes and computed the dissimilarity between the two shapes as a sum of matching errors between corresponding points, together with a term

measuring the magnitude of the aligning transform. The authors have treated recognition in a nearest-neighbor classification framework as the problem of finding the stored prototype shape that is maximally similar to that in the image. They have presented results for silhouettes, trademarks, handwritten digits, and the COIL dataset.

2.1.4. Handwritten Digit Classification using the MNIST Data Set

In this report, the authors have trained and tested a set of classifiers (kth Nearest Neighbor, Gaussian Mixture models and Support Vector Machine) for pattern analysis in solving handwritten digit recognition problems, using the MNIST database. They have concluded that the best results are potentially attainable when the classifiers are combined together.

2.2 Techniques Used

This section gives information about the techniques used in the project.

2.2.1. High quality convolutions-based algorithm with flexible kernel Image Resizing

PIL (Python Imaging Library) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. However, its development has stagnated, with its last release in 2009. Fortunately, there is Pillow, an actively developed fork of PIL, that is easier to install, runs on all major operating systems, and supports Python 3. The library contains basic image processing functionality, including point operations, filtering with a set of built-in convolution kernels, and color-space conversions which are used for image resizing.

2.2.2. Stratified Shuffle Split

Stratified Sampling is a sampling technique that is best used when a statistical population can easily be broken down into distinctive sub-groups. Then samples are taken from each sub-groups based on the ratio of the sub-groups size to the total population. Using the dogs example again, imagine now we have four distinctive breeds of dogs in the 1000 population broken down into A:450, B:250, C:200 and D:100. To perform the sampling of the 200 dogs needed, 45% of the sample must come from A, 25% from B, 20% from C and lastly 10% must come from D. Using Stratified Sampling technique ensures that there will be selection from each sub-groups and prevents the chance of omitting one sub-group leading to sampling bias.

Cross-validation is applied to split the dataset into training and testing sets, retaining 40% of the data for testing. The **StratifiedShuffleSplit** module of the **scikit-learn** Python library was used, passing in the label values as one of the parameters. **StratifiedShuffleSplit** returns train and test indices to split the data into train and test sets while preserving the percentage of the sample of each class.

2.2.3. Principal Component Analysis(PCA)

PCA is a technique for reducing the dimensionality of large datasets and extracting only the relevant features, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. Finding such new variables, the principal components, reduces to solving an eigenvalue/eigenvector problem, and the new variables are defined by the dataset at hand, not *a priori*, hence making PCA an adaptive data analysis technique. It is adaptive in another sense too, since variants of the technique have been developed that are tailored to various different data types and structures. This method was introduced by Karl Pearson.

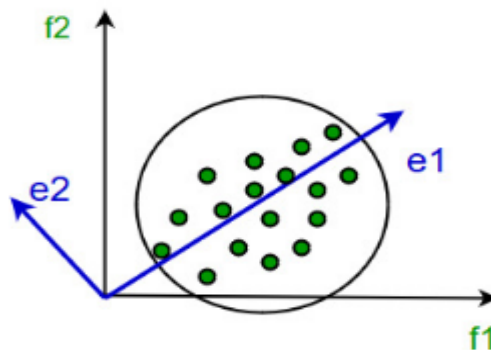


Fig. 2.1. Dimensionality Reduction using PCA

It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum. It involves the following steps:

- Construct the covariance matrix of the data.
- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.

Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.

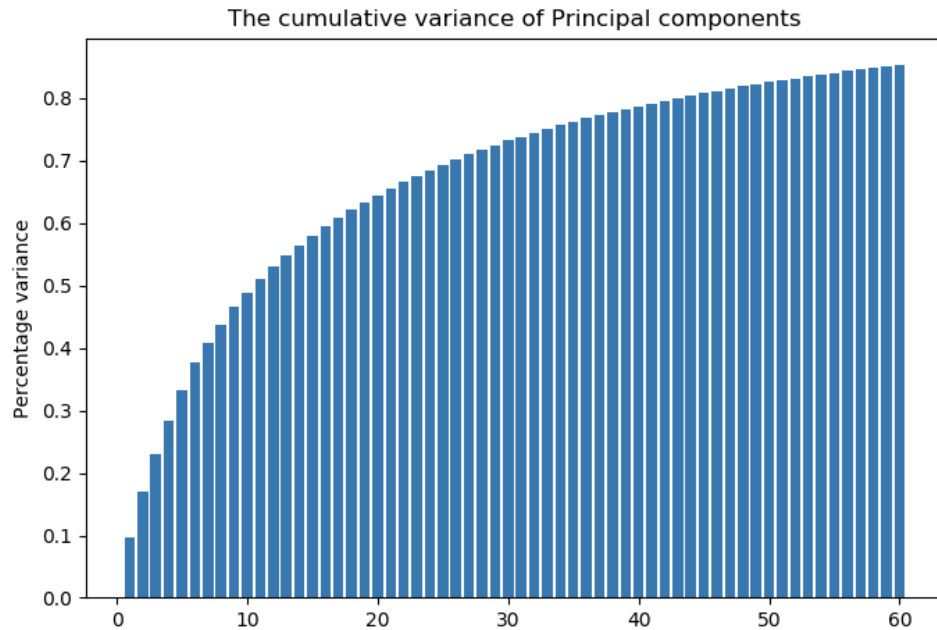


Fig. 2.2. Variance of Principal Components

A 28 X 28=784 pixels image is considered as an input on which the algorithm is to be applied. Since the original dimension is quite large (784 input features), the dimensionality reduction becomes necessary. First, the principal components are extracted from the original data. This is done by fitting a Principal Component Analysis (PCA) on the training set, then transforming the data using the PCA fit. The PCA module of the scikit-learn Python library with **n_components** set to **60** to transform the dataset is used. As shown in the figure 2.3. below, the first **60** principal components can interpret approximately 88% of total information (in terms of total variance retained), which suffice to be representative of the information in the original dataset. We thus choose the first 60 principal components as the extracted features.

```
C:\Users\ashish agarwal\Desktop\pavan imp videos\MPR Project>python gui.py
0.882787002015318
```

Fig. 2.3. Information interpreted by PCA

2.3 Requirements

Requirement analysis results in the specification of software's operational characteristics indicating software's interface with other system elements and establishing constraints that software must meet. Requirement analysis allows the software engineer (sometimes called Analyst or Modeler in this role) to elaborate on basis requirements during earlier requirement engineering tasks and build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior and the flow of data as it is transformed.

The requirements analysis task is a process of discovery, refinement, modeling and specification. The scope, initially established by us and refined during project planning, is refined in details. Model of the required data, information and control flow and operations behavior are created.

2.3.1. Functional Requirements

The functional requirements describe what system has to do.

1. The developed system should recognize handwritten English characters present in the image.
2. System shall show the error message to the user when given input is not in the required format.
3. System must provide the quality of service to the user.
4. System must provide decent accuracy for character recognition.
5. The application should have a graphical user interface.
6. Input of characters with various font sizes and styles should be recognized.
7. The application should identify computer based English characters or digits by comparison.

2.3.2. Non-functional Requirements

As the name suggests, these are the requirements that are not directly interacted with specific functions delivered by the system.

- **Performance:** Handwritten characters in the input image will be recognised with an accuracy greater than 60%.

- **Functionality:** This software will deliver on the functional requirements.
- **Availability:** This system will retrieve the handwritten text regions only if the image contains written text in it.
- **Flexibility:** It provides the users an interface to draw a character or load the image easily.
- **Learn ability:** The software is very easy to use and reduces the learning work.

2.3.3. System Requirements

- **Hardware Requirements:**
 - Intel i3 Processor.
 - 128 MB RAM.
 - 10 GB Hard Disk.
- **Software Requirements**
 - Any Operating System
 - Language: Python 3.x
 - Required Python Library Modules:
 - numpy
 - matplotlib
 - pandas
 - PIL
 - tkinter
 - collections
 - sklearn
 - random
 - time
 - io
 - os

2.3.4. Dataset Requirements

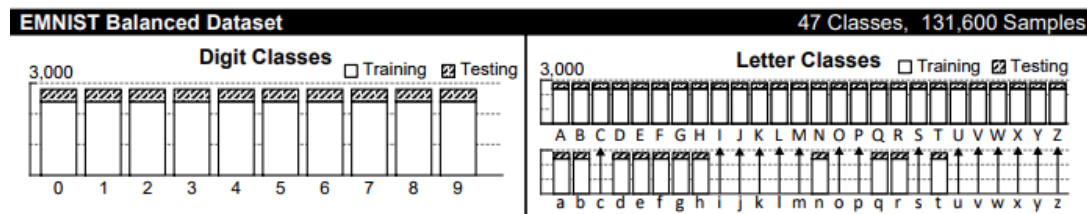


Fig. 2.4. Visual Breakdown of EMNIST Balanced Dataset

The EMNIST Balanced dataset is meant to address the balance issues in the ByClass and ByMerge datasets. It is derived from the ByMerge dataset to reduce mis-classification errors due to capital and lowercase letters and also has an equal number of samples per class. This dataset is meant to be the most applicable.

- train: 112,800
- test: 18,800
- total: 131,600
- classes: 47 (balanced)

This dataset stands as a requirement for initial training of the algorithm working in the application of character recognition system.

2.4 Design

Software Design:

The software has the following design:

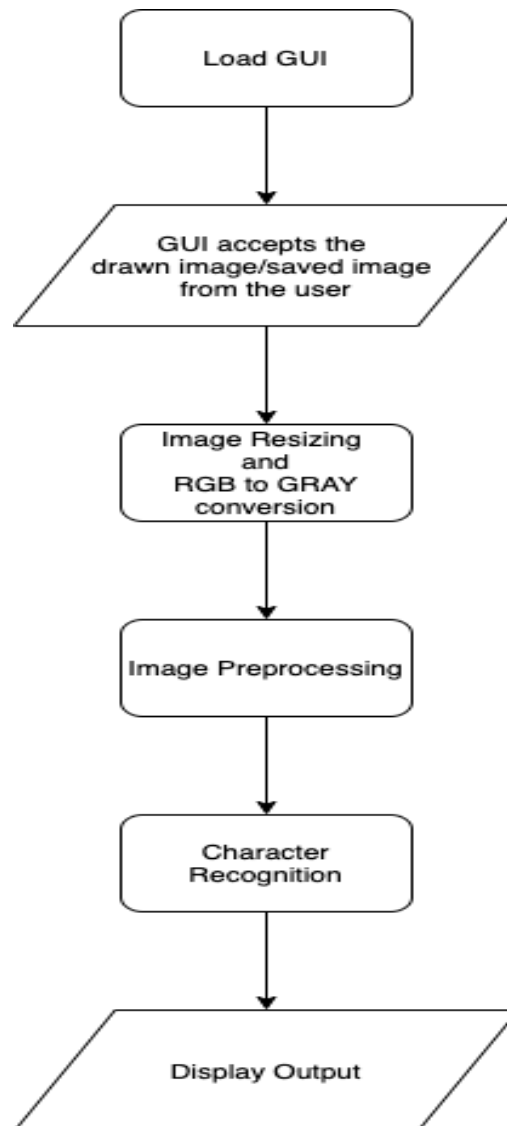


Fig. 2.5. Block diagram of Software Design

➤ **Load GUI:**

- When the program begins execution, this step is first performed.
- In this step, GUI is loaded and canvas is constructed for the image.

- **GUI accepts the drawn image/saved image from the user :**
 - In this step, the input image is acquired from the user.
 - If the image is drawn by the user on the canvas, then it is saved on the external hard disk and then loaded in the main memory for further processing by the program.
 - If the image is uploaded by the user, it is acquired for further processing by the program.
- **Image Resizing and RGB to Gray conversion:**
 - In this step, image is resized to 28 x 28 pixels and converted from rgb image to grayscale image by Pillow.
- **Image Preprocessing:**
 - In this step ,image is processed and features are reduced from 784 to 60.
- **Character recognition:**
 - In this step, the character in the image is recognized using the SVM-KNN combo algorithm.
- **Display Output:**
 - Optically Recognized Characters i.e resultant output of the algorithm is mapped and displayed on the screen.

3. IMPLEMENTATION

3.1 Algorithm Used

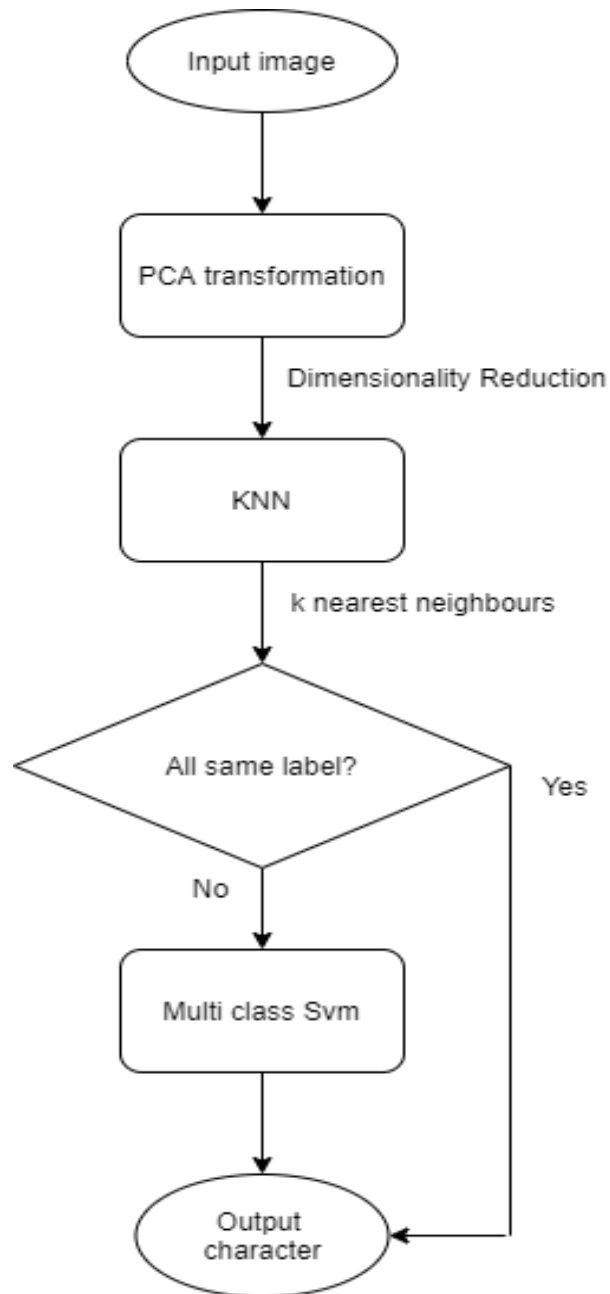


Fig 3.1 Algorithms Functioning

KNN(K-nearest neighbours)

In pattern recognition, the **k-nearest neighbors algorithm (k-NN)** is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the **feature space**. The output depends on whether k -NN is used for classification or regression:

- ❖ In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive **integer**, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.
- ❖ In *k-NN regression*, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

k -NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation.

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for k -NN classification) or the object property value (for k -NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the k -NN algorithm is that it is sensitive to the local structure of the data.

K-nearest neighbor or *kNN classification* determines the decision boundary locally. For 1NN we assign each document to the class of its closest neighbor. For kNN we assign each document to the majority class of its k -closest neighbors where k is a parameter. The rationale of kNN classification is that, based on the contiguity hypothesis, we expect a test document 'd' to have the same label as the training documents located in the local region surrounding 'd'.

Decision boundaries in 1NN are concatenated segments of the *Voronoi tessellation* as shown in Figure[7]. The Voronoi tessellation of a set of objects decomposes space into Voronoi cells, where each object's cell consists of all points that are closer to the object than to other objects. In our case, the objects are documents. The Voronoi tessellation then partitions the plane into $|D|$ convex polygons, each containing its

corresponding document (and no other) as shown in Figure[7] , where a convex polygon is a convex region in 2-dimensional space bounded by lines.

For general k in k NN, consider the region in the space for which the set of k -nearest neighbors is the same. This again is a convex polygon and the space is partitioned into convex polygons , within each of which the set of k -nearest neighbors is invariant .

1NN is not very robust. The classification decision of each test document relies on the class of a single training document, which may be incorrectly labeled or atypical. k NN for $k > 1$ is more robust. It assigns documents to the majority class of their k -closest neighbors, with ties broken randomly.

There is a probabilistic version of this k NN classification algorithm. We can estimate the probability of membership in class c as the proportion of the k -nearest neighbors in c . Figure[7] gives an example for $k=3$. Probability estimates for class membership of the star are $p(\text{circle class}|\text{star}) = 1/3$, $p(\text{X class}|\text{star})=2/3$, and $p(\text{diamond class}|\text{star})=0$ The 3nn estimate ($p(\text{circle class}|\text{star}) = 1/3$ and the 1nn estimate ($p(\text{circle class}|\text{star}) = 1$ differ with 3nn preferring the X class and 1nn preferring the circle class .

The parameter k in k NN is often chosen based on experience or knowledge about the classification problem at hand. It is desirable for k to be odd to make ties less likely. $k=3$ and $k=5$ are common choices, but much larger values between 50 and 100 are also used. An alternative way of setting the parameter is to select the k that gives best results on a *held-out* portion of the training set.

SVM(SUPPORT VECTOR MACHINE)

In machine learning, **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. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic **binary** linear classifier[10] (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the [kernel trick](#), implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The **support-vector clustering** algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

Multiclass SVMs

SVMs are inherently two-class classifiers. The traditional way to do multiclass classification with SVMs is to use one of the methods explained in [9]. In particular, the most common technique in practice has been to build C one-versus-rest classifiers (commonly referred to as “one-versus-all” or OVA classification), and to choose the class which classifies the test datum with the greatest margin. Another strategy is to build a set of one-versus-one classifiers, and to choose the class that is selected by the most classifiers. While this involves building $(|C|)(|C|-1)/2$ classifiers, the time for training classifiers may actually decrease, since the training data set for each classifier is much smaller.

However, these are not very elegant approaches to solving multiclass problems. A better alternative is provided by the construction of multiclass SVMs, where we build a two-class classifier over a feature vector $\Phi(X,Y)$ derived from the pair consisting of the input features and the class of the datum. At test time, the classifier chooses the class

$$\underline{y = \arg \max_{y'} \bar{w}^T \Phi(\vec{x}, y')}$$

. The margin during training is the gap between this value for the correct class and for the nearest other class, and so the quadratic program formulation will require that

$$\underline{\forall i \forall y \neq y_i \bar{w}^T \Phi(\vec{x}_i, y_i) - \bar{w}^T \Phi(\vec{x}_i, y) \geq 1 - \xi_i}$$

. This general method can be extended to give a multiclass formulation of various kinds of linear classifiers. It is also a simple instance of a generalization of classification where the classes are not just a set of independent, categorical labels, but may be arbitrary structured objects with relationships defined between them. In the SVM world, such work comes under the label of *structural SVMs*.

3.2 OUTPUT

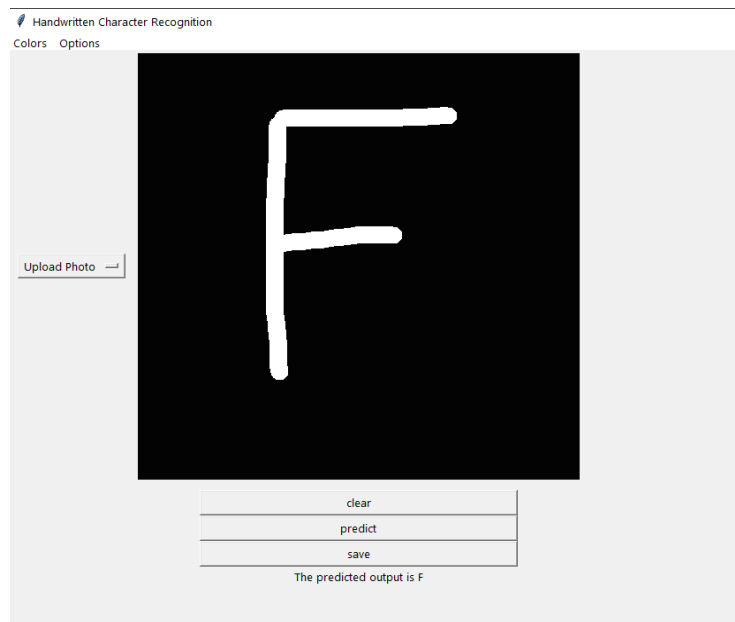


Fig 3.2 Character Prediction 1

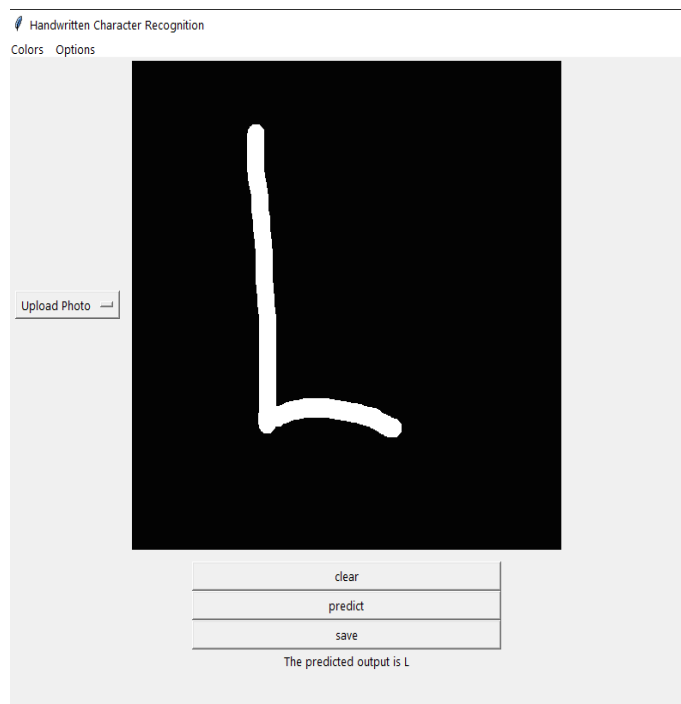
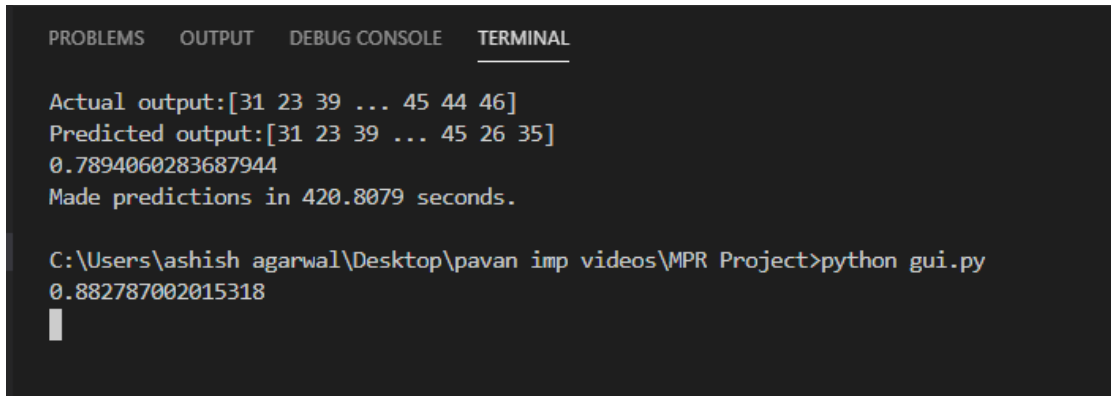


Fig 3.3 Character Prediction 2

3.3 Results

The algorithm SVM-KNN in combination produced the accuracy of 98% accuracy on a test dataset of digit recognition . To improve classification accuracy , techniques like cross validation were used. On character dataset The SVM-KNN algorithm used gave the accuracy of approximately 78%. In digit Recognition 10 classes (0-9) were used for classification. Instead of 47 classes in character recognition. This showed us, the algorithm works well despite significant increase in classes.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Actual output:[31 23 39 ... 45 44 46]
Predicted output:[31 23 39 ... 45 26 35]
0.7894060283687944
Made predictions in 420.8079 seconds.

C:\Users\ashish agarwal\Desktop\pavan imp videos\MPR Project>python gui.py
0.882787002015318
█
```

Fig 3.4 Accuracy on character recognition

The time taken for prediction of 40% of total data (i.e. test data) is 412 seconds in total . on average it takes 0.5-1 seconds for a prediction of single test data . The key to use KNN is , it can be apply to very large dataset with high dimension with varying degree to avoid noisy data and the SVM can be used for small datasets with small dimensions to predict and classify data accurately combining both the aspects our conclusion is To use KNN for coarse categorization and use SVM to perform fine tuning on the image .

4. CONCLUSION AND FUTURE SCOPE

4.1. Limitations of Data:

The training set contained 47 characters with 2500 samples from each character class. The test set contained 47 characters with 500 samples. All the data was obtained from Emnist online website. Also the dataset contains some noise as well as Mirror image, Water image and rotated image of some character. For example 'u' if rotated can be seen as '2' and other characters as shown below.

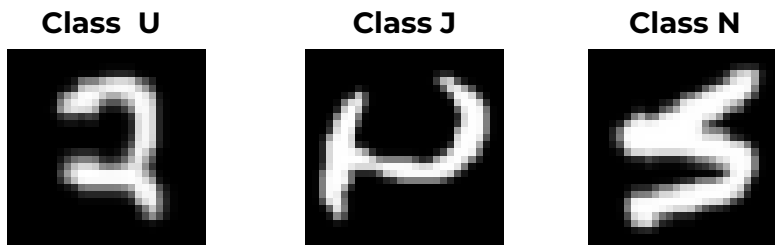


Fig 4.1. Classes similar to 'U'

4.2. Conclusion:

The average running time is greater than Neural Network based approaches in OCR but we can see that instead of a black box kind of nature SVM-KNN algorithm approach is good to interpret the insights, our model is learning.

The overall accuracy of character recognition can be improved by considering only the image that is not water image, mirror image and rotated to more than 45 degrees. The overall accuracy of the system reached is 78%.

4.3. Future Scope:

The work reported in this thesis can be extended in the following directions.

1. Language Converter through OCR

Once a complete OCR has been developed for two languages with font encoding, spell checker and grammatical sentence check, then a converter could be implemented to convert sentences from one language to another through a transliteration and translation scheme.

2. Speech recognition from OCR

The most required application today is Speech recognition. The recognized Printed or Handwritten character could be recorded and through a voice synthesizer speech output could be generated. This would help the blind to send and receive information.

3. A Bilingual or multilingual script OCR

It is basically necessary to develop an OCR for multilingual script for a country where more than 10 languages are in use officially. For example if there is a document where two language scripts are available, then one need not separate two scripts into two different files and feed them into two OCR's. Using our approach one could develop an OCR for two languages, Tamil and English in the same document. These scripts could be edited later with an editor too

4. Speech to text converter through OCR

Similar to the previous application it is possible to convert a speech to text too.

5. LIST OF REFERENCES

1. Classification: https://en.wikipedia.org/wiki/Statistical_classification
2. Pattern recognition: https://en.wikipedia.org/wiki/Pattern_recognition
3. Non-parametric method: https://en.wikipedia.org/wiki/Non-parametric_statistics
4. Regression: https://en.wikipedia.org/wiki/Regression_analysis
5. Instance-based Learning: https://en.wikipedia.org/wiki/Instance-based_learning
6. Lazy learning: https://en.wikipedia.org/wiki/Lazy_learning
7. Rocchio-Classification
<https://nlp.stanford.edu/IR-book/html/htmledition/rochio-classification-1.html#fig:knnboundaries>
8. Support Vector Machines
Ben-Hur, Asa; Horn, David; Siegelmann, Hava; Vapnik, Vladimir N.
""Support vector clustering" (2001);". *Journal of Machine Learning Research*..
9. Classification with more than two classes
<https://nlp.stanford.edu/IR-book/html/htmledition/classification-with-more-than-two-classes-1.html#sec:more-than-two-classes>
10. Probabilistic_classification: https://en.wikipedia.org/wiki/Probabilistic_classification
11. Hao Zhang, Alexander C., Berg Michael, Maire Jitendra
Malik,SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition
http://www.vision.caltech.edu/Image_Datasets/Caltech101/nhz_cvpr06.pdf
12. CHARACTER RECOGNITION IN NATURAL IMAGES
http://personal.ee.surrey.ac.uk/Personal/T.Decampos/papers/decampos_etal_visapp2009.pdf
13. Shape Matching and Object Recognition using Shape Contexts
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.441.6897&rep=rep1&type=pdf>

14. Handwritten Digit Classification using the MNIST Data Set

https://github.com/sbrb/ocr_mnist/blob/master/papers/knn_MNIST.pdf

15. EMNIST Dataset:

<https://arxiv.org/abs/1702.05373v1>

6. ACKNOWLEDGMENT

Our special gratitude to our project guide, Prof. Shilpa Verma for her inspiration , adroit guidance, constant supervision and her short and concise notes in successful completion of the project.

We are very grateful to express our deep sense of gratitude to Dr.Tanuja Sarode ,Professor and Head of Computer Department,Thadomal Shahani Engineering College,Mumbai and for her constant encouragement throughout the project .

We are very thankful to all our Professors: Dr.Tanuja Sarode, Prof Shilpa Verma , Prof Darakshan khan lecturer for providing adequate and prompt facilities during the course of our project.

Our sincere thanks to all Gregory Cohen, Saeed Afshar, Jonathan Tapson, André van Schaik, the creators of EMNIST dataset. This project would not have been achievable without their contribution.

We would also like to thank our family, fellow colleagues and all the people that have supported us.