INTELLIGENT BUS STOP RECOGNITION SYSTEM

A PROJECT REPORT

Submitted by

ATEENDRA R (312213205023)

G GAUTHAM KRISHNA (312213205036)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING,
KALAVAKKAM - 603 110

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2017

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"INTELLIGENT BUS STOP RECOGNITION SYSTEM"** is the bonafide work of **"ATEENDRA R (312213205023)** and **G GAUTHAM KRISHNA (312213205036)"** who carried out the project work under my supervision.

SIGNATURE
Dr. T.NAGARAJAN
HEAD OF THE DEPARTMENT
Professor
Information Technology
SSN College of Engineering
SSN Nagar - 603110.

SIGNATURE
Dr. R. SRINIVASAN
SUPERVISOR
Professor
Information Technology
SSN College of Engineering
SSN Nagar - 603110.

Submitted for the exam to be held on _____

SIGNATURE
INTERNAL EXAMINER

SIGNATURE
EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to our Principal, **Dr. S. Salivahanan**, and our Head of the Department, **Dr. T. Nagarajan**, for having given us the opportunity to carry out this project.

It's our pleasure to thank our Project Guide and Project Coordinator, **Dr. R. Srinivasan**, Professor, Department of Information Technology, for his invaluable guidance and continued support that ensured a smooth progression and a successful completion of our project, and also for providing us with the necessary resources for the completion of our project.

We sincerely thank our project panel members and the various teaching and non-teaching staff of SSN College of Engineering for their help during the various stages of our project.

We would like to take this opportunity to thank our parents, for encouraging and supporting us all through and our friends who have been our pillar of strength during times of difficulty. We also extend our regards to all others who supported us in different ways during the course of our project.

Above all, we thank the Almighty for showering His blessings upon us and making our lives better.

ABSTRACT

Bus Stop Recognition Systems have been widely used till date. These recognition mechanisms have been implemented mainly using a Global Positioning System lookup. The proposed system aims to extract the features of the image of the bus stop/bus station and its surrounding environment, thereby eliminating the need for network. We aim to create this system to achieve high levels of prediction accuracy, by creating a model that learns from a myriad of features. The system utilizes a vision-based approach where the images are received from the cameras placed atop the bus. These images are fed into a light-weight and a simple algorithm to classify the same. However, this project has been implemented on a run-time environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATONS

ABBREVIATION	EXPANSION
GPS	Global Positioning System
GIS	Geographical Information System
k	Number of nearest neighbors
k-NN	kth Nearest Neighbors
RFID	Radio Frequency Identification

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION TO BUS STOP RECOGNITION SYSTEM

Buses are a widespread mode of transportation that are used by the majority of the public. Most buses do not have a system that intimates the passenger that a particular bus stop has arrived. It requires some assistance from another fellow passenger or enquiring the bus driver or conductor. The system proposed in this project would eliminate the need for this enquiry. This would help in a passenger be aware of their location and would not have to depend on a third party. Such a system would also be of much help to the visually impaired community as they face a lot of issues in mobility.

The most used transport means for blind people is the public transportation which is considered as one of the important means for travelling in many countries. Unfortunately, public transportation is not an easy mean to use and access by blind people in many countries. For example, in the case of buses, blind people have difficulty in recognizing bus stops. Unlike normal people who travel independently, blind people need support in guiding them continuously to avoid accidents as well as the unacceptable lateness in their appointments and meetings which may affect their performance as active members in the society. Furthermore, the difficulty of using the public transportation by blind people will make them more isolated and unable to live their normal life.

Such a system would be of great help to foreigners and people who are generally unfamiliar with the route of the bus to become well informed of their commute and be more independent. This system would also relieve the bus

conductors of having to intimate the passengers in case of a new bus stop. Thus, they can focus on their distribution of tickets.

1.2. MOTIVATION

The motivation of building this system is to simulate the thinking of the brain, that is, it was built in a way that it imitates or mimics how a human would react when he/she is asked what the particular bus stop is. In the intelligent pattern recognition, we intend to implement this property. If a human is asked to identify a bus stop, then the individual would look to their left and their right and then make a decision. The ideas that have also been taken into account in this project, is when the system cannot classify the bus stop given only 2 images. So, this motivated the thought of making a more accurate pattern matching system that mines data efficiently. Using the human mind analogy, it was concluded that when the human is unsure of the bus stop, they would look for more images in order to confirm the person questioning them. Such a thought further helped in improving the performance of the system in making more intelligent decisions which in turn would make the system more robust and efficient.

The core recognition module of the system uses a simple k-NN algorithm, i.e. kth Nearest Neighbors after a preprocessing of the image that requires a simple resizing and normalization of the same. The system was also tested using conventional techniques as well as using the intelligent pattern matching techniques which are discussed in later chapters. The preprocessing was designed in order to make the acquired image to be smaller in size thus making storage as well as computation much faster. This further makes the system more hardware-friendly and thus scalable and commercially feasible.

1.3. PROBLEM STATEMENT

To identify bus stops using images acquired from cameras placed on a bus, using a lightweight and a simple algorithm and to explore the efficiency of the algorithm used in terms of both memory and space.

1.4. OBJECTIVE

The objective of the project is to design a Bus Stop Recognition System only using images, that is efficient and accurate, and eliminating the need for sensor networks.

1.5. ORGANIZATION OF REPORT

The detailed organization of the paper is as follows:

Chapter 2 presents the findings of previously conducted research work.

Chapter 3 deals with the Design and flow of the system.

Chapter 4 presents the Training and Testing Datasets.

Chapter 5 discusses about the Implementation and the Software used.

Chapter 6 discusses about Conventional Pattern Recognition Techniques

Chapter 7 deals with the Intelligent Pattern Recognition Technique

Chapter 8 presents the Results and Inferences of various approaches

Chapter 9 concludes the Report along with Future Works.

CHAPTER 2

LITERATURE REVIEW

2.1. SURVEY OF EXISTING SYSTEMS

A Primary Travelling Assistant System of Bus Detection and Recognition for Visually Impaired People

This paper also focuses on an image based approach but it is used for identification of a bus rather than the bus stop itself. It uses HOG (Histogram of the Oriented Gradient) to extract the features of the image obtained. These features are then passed through a Support Vector Machine (SVM) to detect the bus number which in turn would be able to recover the bus route information [1]. However, this employs a HOG model which in turn increases the computational complexity of the system.

Bus Detection System for Blind People using RFID

In this paper, the authors propose a system in which an RFID tag is used to identify the bus stop. The system detects the arrival of a blind person and keeps track of the same. The end-user then uses a tag ID which identifies the bus and gets the bus route. The bus stop identified is announced to the people in the bus [2]. A prototype model of the same has been implemented, tested and validated by the authors.

Recognition of Bus Stops through Computer Vision

This paper introduces a real-time Image Processing System designed for autonomous guidance of passenger buses at bus stops. It continuously estimates the position of the bus relative to a world coordinate system, given by markings, stop

lines or other patterns. The pose estimation is based on the monocular image sequence delivered by a forward-looking camera. The system matches the model of an upcoming bus stop against the grabbed images and estimates the 6 degrees of freedoms of the camera. Cinematic constraints are taken into account by means of a Kalman Filter. Since bus stops do not obey a standard geometry, the models are built from the image sequence directly in a teach-in phase. Although the system is proposed for recognition of a Bus stop, it can be applied to recognize any well-described pattern, e.g. exits on highways or marked parking lots [3].

GPS Based Systems

Normally, a GPS system can be integrated with the bus and whenever the bus arrives at a bus stop, the system proceeds to do a GIS (Geographical Information System) lookup which in turn must be compared with a table consisting of various set of GPS coordinates of bus stops. The closest of all the bus stops would then be displayed to the user.

CHAPTER 3

SYSTEM DESIGN AND FLOW

This chapter deals with the module-wise explanation of each sub-system in the proposed system and the flow of the same.

3.1. VARIOUS MODULES OF THE SYSTEM

The various modules of the system and their corresponding functionalities are as follows.

3.1.1. Image capture of Bus Stops using Cameras

The image is acquired from the bus stop by the cameras present in the system. The data set creation in this project is further explained in the Chapter 4.

3.1.2. Image Normalization and Image Resizing (Preprocessing)

This module deals with the normalization of the image and resizing of the same. The implementation and the corresponding code snippets are present in Chapter 5.

3.1.3. Processing of Images

This is the core module of the algorithm which is implemented in a runtime model which is tested using Conventional and Intelligent Pattern Recognition Techniques which are seen in Chapters 5, 6 and 7.

3.1.4. Database

It is the set of training images that present for all the classes of images. The creation of the database and the specifications of data are given in Chapter 4.

3.2. FLOW OF SYSTEM

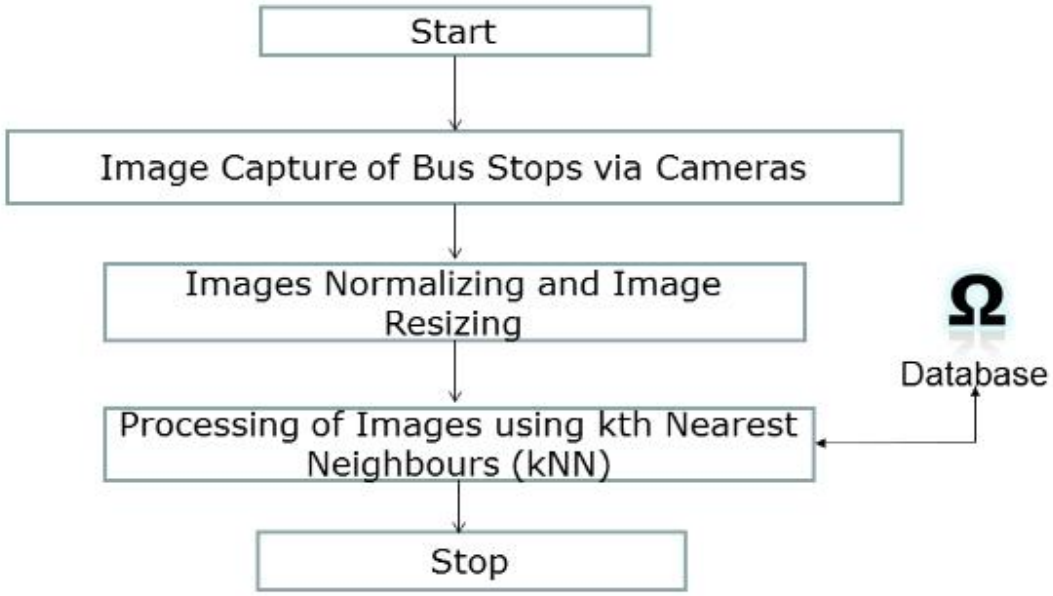


Figure 3.1. Flow of System

CHAPTER 4

DATA ACQUISITION

The data was taken using a Moto G4 Plus phone camera, and a total of 8 bus stops were taken. 5 external bus stops were taken and 3 internal bus stops were taken, as discussed in the tables below.

4.1.1 TRAINING DATA

A total of 720 training images were taken with 360 images on the left side of the bus (main bus stop), and 360 images on the right side of the bus (opposite side of the bus stop).

Table 4.1. Training Dataset

	BUS STOP	**No. of Images**
External	7H	90
	Anna Nagar West Depot	90
	Collector Nagar	90
	Mogappair East	90
	Thirumangalam	90
Internal	Gurunath	90
	Incubation Centre	90
	SOMCA	90
TOTAL		720

4.2. TESTING DATA

A total of 240 training images were taken with 120 images on the left side of the bus (main bus stop), and 120 images on the right side of the bus (opposite side of the bus stop).

Table 4.2. Testing Dataset

	BUS STOP	**No. of Images**
External	7H	30
	Anna Nagar West Depot	30
	Collector Nagar	30
	Mogappair East	30
	Thirumangalam	30
Internal	Gurunath	30
	Incubation Centre	30
	SOMCA	30
TOTAL		240

The training images of all 8 bus stops are stored in a directory with all 720 images in the same directory, since training is a one-time process.

The testing images of the 8 bus stops are stored in separate directories each, with 30 images in each directory (each bus stop), for testing various techniques.

The images are stored in the database, and then accessed via the Jupyter-Notebook (for Python) for running various techniques of k-NN implementation.

4.3. SAMPLE DATASET

Figure 4.1. 7H Bus Stop

Figure 4.2. Anna Nagar West Depot

Figure 4.3. Collector Nagar Bus Stop

Figure 4.4. Mogappair East Bus Stop

Figure 4.5. Thirumangalam Bus Stop

Figure 4.6. Gurunath Bus Stop

Figure 4.7. Incubation Centre Bus Stop

Figure 4.8. SOMCA Bus Stop

4.4. DOWN-SAMPLING OF IMAGES

a. Resolution: 2592 x 4608 x 3

b. Resolution: 400 x 400 x 3

c. Resolution: 300 x 300 x 3

d. Resolution: 200 x 200 x 3

e. Resolution: 100 x 100 x 3

f. Resolution: 32 x 32 x 3

Figure 4.9. Down-Sampling of Images

CHAPTER 5

IMPLEMENTATION AND SOFTWARE

5.1. SOFTWARE USED

5.1.1. Python

Python is a widely used high-level programming language for general purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and a comprehensive standard library.

Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late bidding), which binds method and variable names during program execution.

Python has a set of large standard libraries, commonly cited as one of Python's greatest strengths, providing tools suited for many tasks. Most Python implementations include a read-eval-print loop, which essentially functions as a command-line interpreter.

5.1.2. NumPy

NumPy is the fundamental package for scientific computing with Python. It contains a lot of advantages, a few listed below,

- A powerful N-dimensional array (nd-array) object

- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, sorting, selecting, I/O and random number capabilities
- Basic linear algebra, basic statistical operations, random simulation and much more.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. There are several important differences between NumPy arrays and the standard Python sequences:

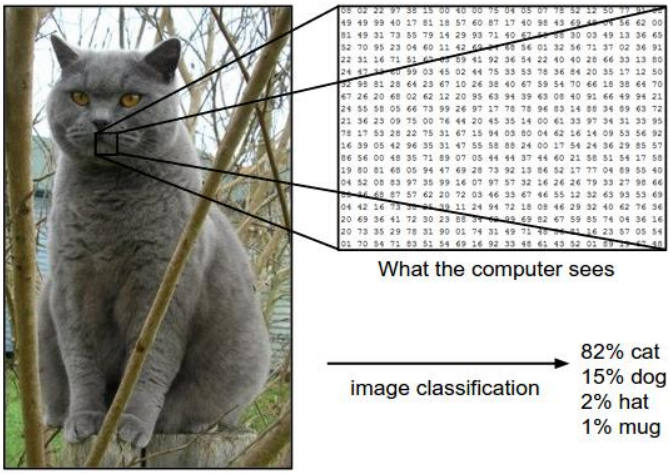
- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an nd-array will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output

NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

5.2. IMAGE CLASSIFICATION

In this section the Image Classification problem is introduced, which is the task of assigning an input image, i.e., one label from a fixed set of categories. This is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. Moreover, many other seemingly distinct Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification.



5.1. A Typical Example of a Pixelated Image

The task in Image Classification is to predict a single label (or a distribution over labels) as shown here to indicate our confidence for a given image. Images are 3-dimensional arrays of integers from 0 to 255, of size Width x Height x 3. The 3 represents the three color channels Red, Green, Blue.

5.2.1. Challenges

Since this task of recognizing a visual concept is relatively trivial for a human to perform, the challenges involved from the perspective of a Computer Vision algorithm should be considered.

Viewpoint variation: A single instance of an object can be oriented in many ways with respect to the camera.

Scale variation: Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).

Deformation: Many objects of interest are not rigid bodies and can be deformed in extreme ways.

Occlusion: The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible.

Illumination Conditions: The effects of illumination are drastic on the pixel level.

Background Clutter: The objects of interest may blend into their environment, making them hard to identify.

Intra-class variation: The classes of interest can often be relatively broad, such as chair. There are many different types of these objects, each with their own appearance.

A good image classification model must be invariant to the cross product of all these variations, while simultaneously retaining sensitivity to the inter-class variations.

5.2.2. Data-Driven Approach

Instead of trying to specify what every one of the categories of interest look like directly in code, the approach that we will take is more like one which we would take with a child: we're going to provide the computer with many examples of each class and then develop learning algorithms that look at these examples and learn about the visual appearance of each class. This approach is referred to as a data-driven approach, since it relies on first accumulating a training dataset of labeled images.

5.2.3. The Image Classification Pipeline

We've seen that the task in Image Classification is to take an array of pixels that represents a single image and assign a label to it. Our complete pipeline can be formalized as follows:

Input: Our input consists of a set of N images, each labeled with one of K different classes. We refer to this data as the training set.

Learning: Our task is to use the training set to learn what every one of the classes looks like. We refer to this step as training a classifier, or learning a model.

Evaluation: In the end, we evaluate the quality of the classifier by asking it to predict labels for a new set of images that it has never seen before. We will then compare the true labels of these images to the ones predicted by the classifier.

Intuitively, we're hoping that a lot of the predictions match up with the true

answers (which we call the ground truth).

```
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```

Figure 5.2. Training by Labeled Classes

5.3. NEAREST NEIGHBOR CLASSIFIER

The Nearest Neighbor Classifier predicts the labels based on nearest images in the training set. The input consists of the k closest training examples in the feature space. The output of the kth Nearest Neighbor (k-NN) is used for classification.

In k-NN classification, the output is a class membership. An object is classified by a majority 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. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and computation is deferred until classification.

The neighbors are taken from a set of objects for which the class is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. k-NN is used in other domains such as NLP, simple Machine Learning problems, etc. It is a category of Instance based learning systems that store the features as such and use distance similarities such as Euclidean/Manhattan distance in case of Euclidean space, Cosine distance, Mahanobolis distance, etc otherwise.

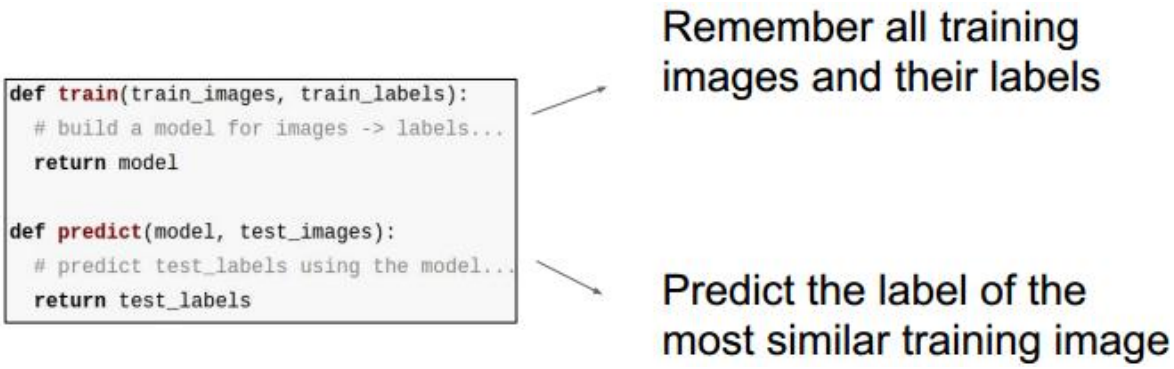


Figure 5.3. Remember and Predict k-NN Model

5.4. IMPLEMENTATION

The Algorithm was implemented using the Jupyter-Notebook step-by-step, which are presented as snippets here.

Import Libraries: Import all the necessary packages like **numpy** (a Python library for high order scientific computation using matrices), **random** (a Python library for random number generation), **matplotlib** (a Python library for plotting and visualizing of data) and **scipy**(a Python library for implementation of scientific computations and resizing).

```
importrandom
importnumpyasnp
importmatplotlib.pyplotasplt
importscipy.miscassc

%matplotlib inline
```

Image Resizing: A function called image_resize () is used to resize the image to the necessary shape for easier computations (different image sizes are taken as parameters and plotted in detail in Chapter 8).

```
defimage_resize(im):
im=sc.imresize(im,(32, 32,3))
returnim
```

Loading Training Data: The training data in k-NNset of all 8 bus stops are loaded from the database using the imread () function from matplotlib, as a numpy array, and they are resized using the image_resize () function to the required shape as mentioned in step 2. The resized image is then flattened as a single vector of data points. This is then converted to the float32 type, and stored in X_train.

```
X_train=np.array(image_resize(plt.imread("./FinalTrainDay/Train1.jpg")).fl
atten().astype("float32"))
foriinrange(2,721):
ifi%25==0:
print("Reading train image "+str(i))
img=plt.imread("./FinalTrainDay/Train"+str(i)+".jpg")
X_train=np.vstack((X_train,image_resize(img).flatten().astype("float32")))
print(len(X_train))
```

Image Normalization: After loading all the training images into X_train, the images are normalized to express the 8-bit RGB to a value between 0-1.Since 255 is the maximum value for a channel, dividing by 255 expresses a 0-1 representation. This is the final normalized X_train.

```
X_train=X_train/255.0
```

Setting Training Labels: Now, the labels of the training images are set for all the 8 bus stops and stored in a numpy array called y_train.

```
y_train=[]
forbus_stopinrange(1,9):
fortrain_imagesinrange(90):
y_train.append(bus_stop)
y_train=np.array(y_train)

print(y_train.shape)
```

Computing Distances from Training and Testing: After normalizing, the testing data, X_test is loaded (explained in the forthcoming chapters). The function

compute_distances () is written to calculate the distance between the training and testing numpy arrays. The distance between the numpy arrays X_train and X_test is calculated using Euclidean distance, where a vectorized method of implementation is used without any looping mechanism. The squared difference between the arrays in Euclidean distance is reframed to be of the form,

$$(a - b)^2 = a^2 - 2 * a * b + b^2$$

for a faster and efficient implementation of calculating distances.

```
defcompute_distances(X_train,X):
X=np.array(X)/255.0
X=X.astype("float32")
num_test=X.shape[0]
num_train=X_train.shape[0]
dists=np.zeros((num_test,num_train))
dists=np.sqrt((np.square(X).sum(axis=1,keepdims=True))-(2 *
X.dot(X_train.T))+(np.square(X_train).sum(axis=1)))
returndists
```

CHAPTER 6

CONVENTIONAL PATTERN RECOGNITION TECHNIQUES

6.1. PATTERN RECOGNITION

Pattern recognition is a branch of machine learning that focuses on the recognition of patterns and regularities in data, although it is in some cases considered to be nearly synonymous with machine learning. Pattern recognition systems are in many cases trained from labeled "training" data (supervised learning), but when no labeled data are available other algorithms can be used to discover previously unknown patterns (unsupervised learning).

6.2. PATTERN RECOGNITION TECHNIQUES

There are various approaches to test the system and its algorithmic efficiency such as probabilistic approaches, clustering approaches, ensemble algorithms etc. But there are 2 major conventional techniques used in recognizing the images from our testing dataset. They are:

1. All-Image approach
2. Random trials based approach

6.3. ALL-IMAGE APPROACH

In this approach we use the testing method in which we train the classifier with the existing training dataset, which consists of 60 images in each of the 8 classes i.e. bus stops. As discussed in k-NN, the training data is simply stored as such along with the class labels as it is from instance based learning. A similar process was employed in the acquisition of the testing data. The test images of the

8 classes which consist of the 30 images present for each bus stop, is used for the determination of the accuracy of the system. This is tested based on comparing the image labels to the test labels that is declared for each of the test images respectively.

6.3.1. Features and Variations in All-Image Approach

Variation of Parameter k

The parameter is used to find the k closest images to the test image and also computes the most common of the lot, and in case of a tie, the lowest class number is chosen in each iteration. This further varies the behavior of accuracy of the classifier. This can be observed in the various graphs explained in the further sections.

Size of Image

This varies the resolution of the image, thus leads to change in accuracy. However, lower resolution would lead to loss in information, thus also leads to loss in features received from each image. But, choosing an image of lower resolution helps in reduction of the image size, thus we can store the training dataset in lower resolution. However, increase in resolution is a double-edged sword as we might be observing more features that might be extremely specific thus might result in over fitting.

Time Taken for each k

Since increase in the k parameter will result in more computation, and this time is also tracked in this run-time implementation of the same.

6.3.2. Snippet for All-Images Approach

```
fromtimeimporttime
fromcollectionsimportCounter

x=0
X_test=np.zeros((240,np.prod(X_train.shape[1:])))
foriinrange(1,31):
        X_test_temp=image_resize(plt.imread("./FinalTestDay/7H/Test"+str(i)
+".jpg")).flatten()
        X_test[x,:]=X_test_temp
        x+=1
foriinrange(1,31):
        X_test_temp=image_resize(plt.imread("./FinalTestDay/AN West
Depot/Test"+str(i)+".jpg")).flatten()
        X_test[x,:]=X_test_temp
        x+=1
foriinrange(1,31):
        X_test_temp=image_resize(plt.imread("./FinalTestDay/Coll/Test"
        +str(i)+".jpg")).flatten()
        X_test[x,:]=X_test_temp
        x+=1
foriinrange(1,31):
        X_test_temp=image_resize(plt.imread("./FinalTestDay/Mogs/Test"
        +str(i)+".jpg")).flatten()
        X_test[x,:]=X_test_temp
        x+=1
foriinrange(1,31):
        X_test_temp=image_resize(plt.imread("./FinalTestDay/Thiru/Test"
        +str(i)+".jpg")).flatten()
        X_test[x,:]=X_test_temp
        x+=1
foriinrange(1,31):
        X_test_temp=image_resize(plt.imread("./FinalTestDay/Gurunath/
        Test"+str(i)+".jpg")).flatten()
        X_test[x,:]=X_test_temp
        x+= 1
foriinrange(1,31):
        X_test_temp=image_resize(plt.imread("./FinalTestDay/Incubation
        Centre/Test"+str(i)+".jpg")).flatten()
        X_test[x,:]=X_test_temp
        x+=1
foriinrange(1,31):
        X_test_temp=image_resize(plt.imread("./FinalTestDay/SOMCA/Test"
        +str(i)+".jpg")).flatten()
        X_test[x,:]=X_test_temp
        x+=1

printX_test.shape

dis=compute_distances(X_train,X_test)

forkinrange(1,6):
```

```
        t1=time()
        count=0
        correct_classes=[]
        foriinrange(dis.shape[0]):
                l=y_train[np.argsort(dis[i,:])].flatten()
                closest_y=l[:k]
                correct_classes.append(Counter(closest_y).
                most_common(1)[0][0])
        correct_classes=np.array(correct_classes)
        print"Closest 10 images : "+str(l[:10])
        correct_classes_final=np.sum([correct_classes==y_test])
        print("K          =          "+str(k)+";          Accuracy          :
        "+str(float(correct_classes_final)/dis.shape[0]))
        print
t2=time()
print(t2-t1)
```

- In this snippet, the loading of the training data set is done as said in previous chapters, and is stored in the nd-array, i.e. a n-dimensional array by the name of X_train which is of shape 720 x (image_size * image_size*3) as there are 3 channels (RGB).
- X_test is used to store the test images, as the test dataset consists of 240 images, each class containing about 30 images. All the 240 images are resized and loaded into the respective arrays.
- For both X_train and X_testare normalized and resized as a part of the preprocessing phase. The images are converted.
- The distance matrix namely dis, is computed in order to find the differences between each train example and each test example.
- The k value is varied between from 1-5, to determine the behavior of accuracy with respect to k. Hence there is a loop from 1 to 6, and each value is plugged in and the corresponding results were observed.
- For a sanity check, the 10 closest images' classes are also printed in order to infer the closest images which in turn results in understanding the efficiency of the algorithm.

- Accuracy for each value of k is calculated as:

  $$\text{Accuracy} = (N_C / N_T) * 100$$

  where $\mathbf{N_C}$ **-** Number of correctly classified test images

  $\mathbf{N_T}$**-** Total number of test images

- This computed accuracy is then printed to the user.

6.3.3. Pitfalls of the Approach

Although this approach is widely used to evaluate many models, it might not be the most efficient way of testing this system. It requires the model to be tested such that the images are fed in by batches and the we see the performance of the classifier on this and furthermore calculate the efficiency based on this measure. This discrepancy was noticed and therefore the same in the further testing approaches.

6.4. RANDOM TRIALS APPROACH

In this approach, we use randomized method to evaluate the system's working. The training of the system is implemented as its preceding methods. However, the testing works as a system that sits atop the existing k-NN algorithm which is used for 2 levels of randomization. It was also seen that upon using randomization, any sort of statistical interference would be nullified, thus the results obtained would in turn be uniformly distributed across the classes. This testing technique's code and working are discussed further.

6.4.1. Features and Variations of Random Trials Approach

Variation of Parameter k

Similar to the previous method, the k parameter is varied in this approach in order to classify the image.

Number of Images tested in a trial

The number of images are also varied in order to test the system in order to check the efficiency of the system under increase in the number of test images. This goes to determine the scalability of the system under increase of images which in turn causes an increase in memory and the time taken for the execution of the trial.

Memory Consumed and Time Taken

The memory consumed for the process was also observed and noted down in order to infer the ideal parameter of the system.

6.4.2. Snippet for Random Trials Approach

```
forn_imagesin[2,3,5,7,9,11]:
        input()
        t1=time()
        foriterxinrange(100):
        print"Iter : "+str(iterx)
        class_name=np.random.randint(1,9)
        printclass_name,n_images
        X_test=np.zeros((n_images,3000000))
        x=0
        ifclass_name==1:
        test_images=np.random.randint(1,31,size=n_images)
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTest/7H/Test"
                +str(i)+".jpg")).flatten()
                X_test[x,:]=X_test_temp
                x+=1
        elifclass_name==2:
        test_images=np.random.randint(1,31,size=n_images)
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTest/AN West
                Depot/Test"+str(i)+".jpg")).flatten()
                X_test[x,:]=X_test_temp
```

```
            x+=1
        elifclass_name==3:
        test_images=np.random.randint(1,31,size=n_images)
        foriintest_images:
            X_test_temp=image_resize(plt.imread("./FinalTest/Coll/Test"
            +str(i)+".jpg")).flatten()
            X_test[x,:]=X_test_temp
            x+=1
        elifclass_name==4:
        test_images=np.random.randint(1,31,size=n_images)
        foriintest_images:
            X_test_temp=image_resize(plt.imread("./FinalTest/Mogs/Test"
            +str(i)+".jpg")).flatten()
            X_test[x,:]=X_test_temp
            x+=1
        elifclass_name==5:
        test_images=np.random.randint(1,31,size=n_images)
        foriintest_images:
            X_test_temp=image_resize(plt.imread("./FinalTest/Thiru/
            Test"+str(i)+".jpg")).flatten()
            X_test[x,:]=X_test_temp
            x+=1
        elifclass_name==6:
        test_images=np.random.randint(1,31,size=n_images)
        foriintest_images:
            X_test_temp=image_resize(plt.imread("./FinalTest/Gurunath/
            Test"+str(i)+".jpg")).flatten()
            X_test[x,:]=X_test_temp
            x+=1
        elifclass_name==7:
            test_images=np.random.randint(1,31,size=n_images)
            foriintest_images:
            X_test_temp=image_resize(plt.imread("./FinalTest/Incubation
            Centre/Test"+str(i)+".jpg")).flatten()
            X_test[x,:]=X_test_temp
            x+=1
        elifclass_name==8:
        test_images=np.random.randint(1,31,size=n_images)
        foriintest_images:
            X_test_temp=image_resize(plt.imread("./FinalTest/SOMCA/
            Test"+str(i)+".jpg")).flatten()
            X_test[x,:]=X_test_temp
            x+=1
        dis=compute_distances(X_train,X_test)
        forkinrange(1,6):
            count=0
            correct_classes=[]
            foriinrange(dis.shape[0]):
                l=y_train[np.argsort(dis[i,:])].flatten()
                closest_y=l[:k]
                correct_classes.append(Counter(closest_y).
                most_common(1)[0][0])
            correct_classes=np.array(correct_classes)
            print(l[:10])
```

```
                forvinrange(correct_classes.shape[0]):
                        ifcorrect_classes[v]==class_name:
                                count+=1
                printcorrect_classes
                accuracy=float(count)/dis.shape[0]
                print"K = "+str(k)+"; Accuracy : "+str(accuracy)
        print

t2=time()
printt2-t1
```

- The training data is initially loaded into the system and the preprocessing is done by resizing and normalizing the images and conversion into an Numpy nd-array.
- Now, for each trial, we choose any of the classes (1-8) randomly and this is the initial level of randomization.
- For the class chosen in the previous step, we choose 5 images randomly and we classify the same using the k-NN algorithm.
- If all the 5 images in this trial is classified as the right class, then we consider it as a successful trial if the all the images are classified as the right image.
- The trial is deemed unsuccessful even if any one of the images in the trial is classified incorrectly.
- The accuracy is calculated for the method as follows:

$$\text{Accuracy} = (N_{Success}/N_{Total})*100$$

where, $\mathbf{N_{Success}}$- Number of successful trials

$\mathbf{N_{Total}}$-Total number of trials

- The results of this model are seen in the Results and Analysis chapter.

6.4.3. Pitfalls of Random Trials Approach

This method, despite its efficiency does not correlate with the initial premise that the system was built based on the thinking of the human brain. That is, a human might be wrong on classifying 1 image incorrectly, but they might be able to identify with an assurance in the result that they give. This approach does not take into account this notion. This has been incorporated to an extent in the next chapter on Intelligent Pattern Recognition techniques.

CHAPTER 7

INTELLIGENT PATTERN RECOGNITION TECHNIQUE

As mentioned in the previous chapter, the major shortcoming of the conventional approach is the method's inability to mimic the human mind's working. Thus in this chapter, this problem is tried to be resolved by the use of a better and intelligent pattern recognition mechanism. The motivation behind reducing the image sizes in this technique is that, when one cannot achieve a result by quality, one can do so using the numbers, which might turn the tides in its favor.

7.1. THE CONCEPT OF DECISIONS

So far in the previous methods of testing, we have employed the recognition accuracy as a metric to judge the efficient functioning of the system. But in a real world scenario, we cannot employ a system that discards a set of images in a trial if a single image is misclassified. Thus, we propose the concept of the classifier which would also work as a decision making unit. This was motivated from the idea of simulating a human mind's working. For example, if an individual is asked to identify a bus-stop, the individual would only look for information only if they are unsure of the current information they have at hand. Here, the underlying assumption is that the individual would look for 2 images initially. So, this gives rise to 2 major cases as below:

Case1: The individual can confidently decide on the bus stop using the 2 images only, which may or may not be right.

Case 2: The individual is unsure and requires more information.

Case 2.1: The individual is able to arrive at a confident decision using the existing and new information.

Case 2.2: The individual is unable to arrive at a decision on which bus stop it is.

7.2. IMPLEMENTATION OF INTELLIGENT APPROACH

7.2.1. Features and Variations of Intelligent Approach

- For this approach, we use the initial images as 2, as the proposed system is said to capture images using cameras present on either side of the system.
- The k parameter is set to 1, as the results of the previous methods indicate that k = 1 results in the best possible recognition accuracy.
- The memory is taken into account in terms of the size of the image which was varied from 32x32, 100x100, 200x200, 300x300, 400x400 in order to make the system more hardware friendly.
- The approach strives to increase the accuracy of the decision made to tend to 100%.

7.2.2. Snippet of Intelligent Approach

```
forn_imagesin [2]:
    t1 =time()
foriterxinrange(100):
       print"Iter : "+str(iterx)
       class_name=np.random.randint(1, 9)
       print"Class   name   :   "+str(class_name),   "No.   of   Images   :
       "+str(n_images)
       X_test=np.zeros((n_images, np.prod(X_train.shape[1:])))
       x =0
       ifclass_name==1:
       test_images=np.random.randint(1, 31, size=n_images)
       foriintest_images:
              X_test_temp=image_resize(plt.imread("./FinalTestDay/7H/
              Test"+str(i) +".jpg")).flatten()
              X_test[x, :] =X_test_temp
                x +=1
       elifclass_name==2:
       test_images=np.random.randint(1, 31, size=n_images)
```

```
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTestDay/AN  West
                Depot/Test"+str(i) +".jpg")).flatten()
                X_test[x, :] =X_test_temp
                x +=1
        elifclass_name==3:
        test_images=np.random.randint(1, 31, size=n_images)
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTestDay/Coll/
                Test"+str(i) +".jpg")).flatten()
                X_test[x, :] =X_test_temp
                x +=1
        elifclass_name==4:
        test_images=np.random.randint(1, 31, size=n_images)
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTestDay/Mogs/
                Test"+str(i) +".jpg")).flatten()
                X_test[x, :] =X_test_temp
                x +=1
        elifclass_name==5:
        test_images=np.random.randint(1, 31, size=n_images)
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTestDay/Thiru/
                Test"+str(i) +".jpg")).flatten()
                X_test[x, :] =X_test_temp
                x +=1
        elifclass_name==6:
        test_images=np.random.randint(1, 31, size=n_images)
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTestDay/
                Gurunath/Test"+str(i) +".jpg")).flatten()
                X_test[x, :] =X_test_temp
                x +=1
        elifclass_name==7:
        test_images=np.random.randint(1, 31, size=n_images)
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTestDay/
                Incubation Centre/Test"+str(i) +".jpg")).flatten()
                X_test[x, :] =X_test_temp
                x +=1
        elifclass_name==8:
        test_images=np.random.randint(1, 31, size=n_images)
        foriintest_images:
                X_test_temp=image_resize(plt.imread("./FinalTestDay/SOMCA/
                Test"+str(i) +".jpg")).flatten()
                X_test[x, :] =X_test_temp
                x +=1

         dis =compute_distances(X_train,X_test)

        for kinrange(1, 2):
                count =0
                correct_classes= []
                foriinrange(dis.shape[0]):
```

```
                        l =y_train[np.argsort(dis[i, :])].flatten()
                        closest_y= l[:k]
                        correct_classes.append(Counter(closest_y).
                        most_common(1)[0][0])
                correct_classes=np.array(correct_classes)
                print"Closest 10 images : "+str(l[:10])
                print correct_classes

        forvarinrange(1, 11):
                print"Accuracy : "+str(Counter(correct_classes).
                most_common(1)[0][1] /float(len(correct_classes)))
                ifCounter(correct_classes).most_common(1)[0][1]
/float(len(correct_classes)) >0.5:
                        print"Predicted as ",
                        printcorrect_classes,
                        print"Groundtruth : "+str(class_name)
                        break
                else:
                        print"Fetching New Image ",
                        new_image_no=np.random.randint(1, 31, size=1)
                        whilenew_image_nointest_images:
                        new_image_no=np.random.randint(1, 31, size=1)
                        printtest_images, new_image_no
                        list(test_images).append(new_image_no)
                        new_test=get_one_image(class_name=class_name,
                        rand_num=new_image_no)
                        X_test=np.vstack((X_test, new_test))
                        printX_test.shape
                        dis_new=compute_distances(X_train, X_test)
                        correct_classes1 = []
                        for ii inrange(dis_new.shape[0]):
                      l1 =y_train[np.argsort(dis_new[ii, :])].flatten()
                      closest_y1 = l1[:k]
        correct_classes1.append(Counter(closest_y1).
                        most_common(1)[0][0])
                        ifCounter(correct_classes1).most_common(1)[0][1]
                        /float(len(correct_classes1)) <0.5:
                        print"Back again "+str(Counter(correct_classes1).
                        most_common(1)[0][1] /float(len(correct_classes1)))
                        print"Predicted as : ",
                        print correct_classes1,
                        print"Ground Truth : "+str(class_name)
                        else:
                        print"Predicted as : ",
                        print correct_classes1,
                        print"Ground Truth : "+str(class_name)
                        print"Accuracy now is : "+str
                        (Counter(correct_classes1).most_common(1)[0][1]
                        /float(len(correct_classes1)))
                        break
                print
        t2 =time()
        printt2-t1
```

- The training images are preloaded in the system and they are also preprocessed as mentioned in the previous approaches.
- Even in this implementation, there are 2 levels of randomness over 100 trials in order to bring probabilistic uniformity. These 2 levels of randomness are in the selection of the class and the test images to be fetched from the test dataset.
- These 2 fetched images are preprocessed and they are stored in a Numpy array.
- The decision is made when the majority of the images in the test set belongs to the same class.
- In case a decision is not made, then the system acquires another random image from the test dataset, and the previous step is executed.
- In the extreme case, when the system is not able to arrive at a decision, that is, a majority is not achieved, then in an ideal scenario, it must iteratively acquire more images until the depletion of the test data set from the class. But such an approach would increase the time and memory required for the execution, thus we choose an upper limit of 10 was chosen.
- In the case of a trial in which the system was not able to decide even after 10 of these iterations, it concludes that the decision would not be taken for that particular trial, thus leading to system not announcing anything to the user, just a human would not be able to answer if they did not know the particular bus-stop.
- The results of this implementation is explained in the Results chapter.

CHAPTER 8

RESULTS AND INFERENCES

This section of the report consists of the results and their inferences that were acquired from each of these above pattern recognition techniques.

8.1. RESULTS OF ALL-IMAGE APPROACH

Parameter k vs Accuracy

In this graph 8.1, we plot the accuracy of the system against the parameter k from the k-NN algorithm. This is plotted for various sizes of the images thus resulting in 5 lines as shown in the legend. It is observed that the maximum accuracy was achieved for k = 1and the accuracy was declining. This can be attributed to the fact that as k increases, the tendency for the correct classification of the image goes down due to statistical reasons or even noise that might cause similarity in images from different classes.

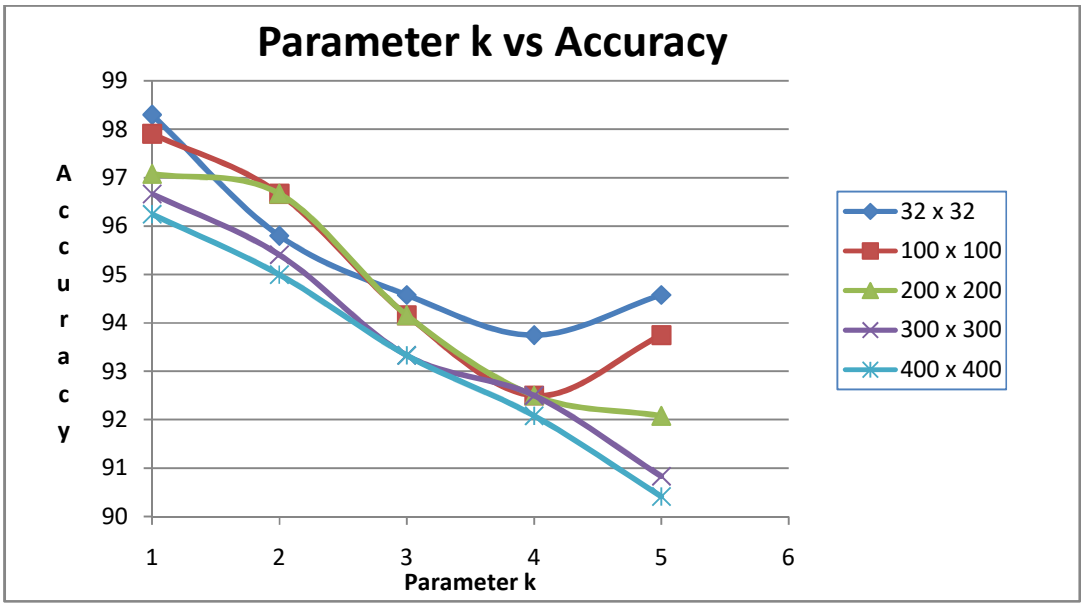


Figure 8.1. Parameter k vs Accuracy Graph

Average Time Taken vs Image Size

In this graph 8.2, the average time taken for each k from 1 to 5 is plotted against image size. We can see that as the size of the image increases the time taken increases due to increase in computation in the distance matrix computation despite its vectorized implementation. The sharp increase shows that as the size of the image increases, the time taken for classification increases rapidly.

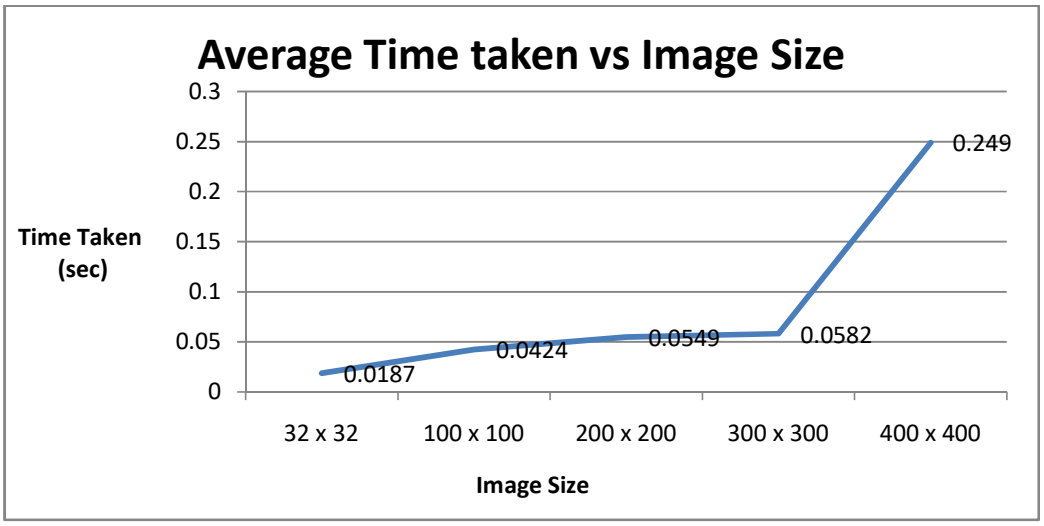


Figure 8.2. Average Time Taken vs Image Size Graph

8.2. RESULTS OF RANDOM TRIALS APPROACH

This approach was implemented in 2 sub-approaches, the specifications for them are as follows:

Sub-Approach 1

Only 5 classes of images, pertaining to the external bus-stops were considered and the k parameter was set to 1 due to memory constraints.

Sub-Approach 2

8 classes were considered which includes both internal and external bus-stops were used and the code was slightly tweaked in order to more efficient in terms of memory and space.

8.2.1. Results of Sub-Approach 1

Accuracy vs Number of Images

In this graph 8.3, the accuracy of the classifier is plotted against the number of images which are varied from 2-11. The resulting bump in the middle of the curve could be because of the superimposition of 2 learning curves, which is the monotonically decreasing one and a bitonic curve, thus resulting in the graph shown below.

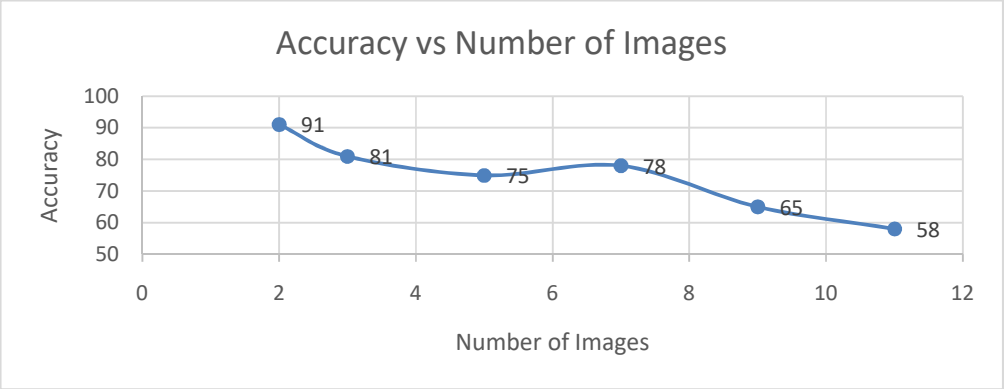


Figure 8.3. Accuracy vs Number of Images

Maximum Memory Consumed and Time consumed vs Number of Images

These graphs 8.4 and 8.5, show the relationship between the maximum memory consumed in GB and the time taken for the same with the number of images. It is a straightforward conclusion that memory consumed increases with the number of images.

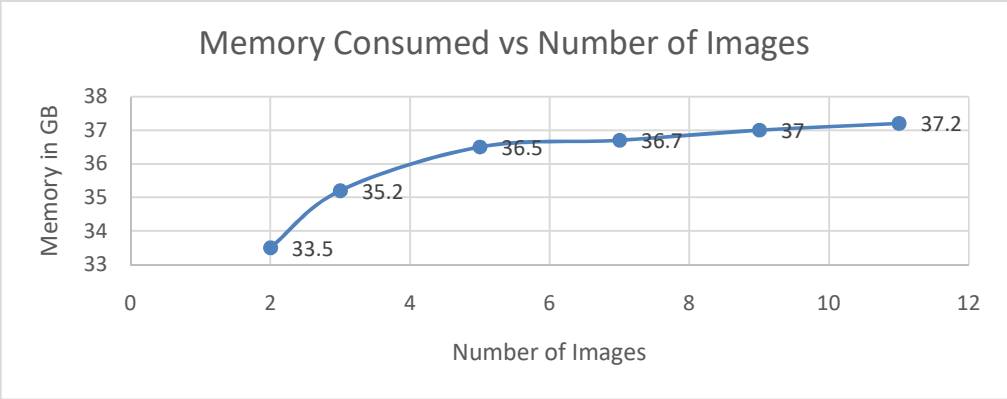


Figure 8.4. Memory Consumed vs Number of Images Graph

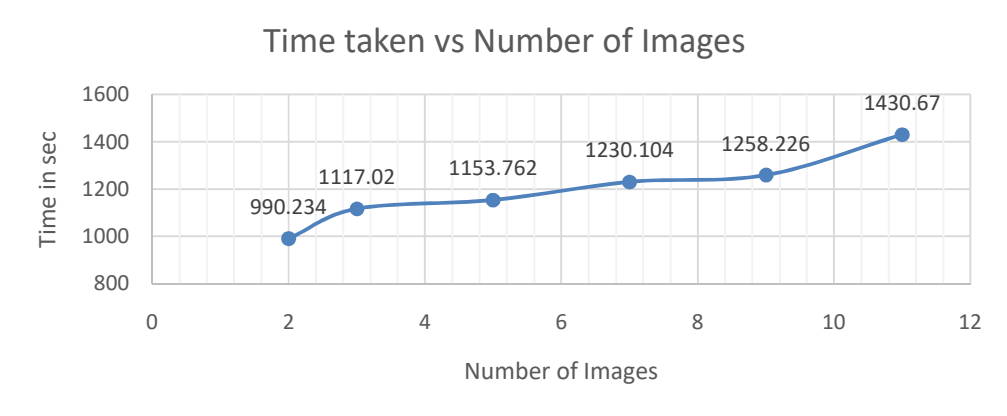


Figure 8.5. Time Taken vs Number of Images Graph

8.2.2. Results of Sub-Approach 2

Accuracy vs Parameter k and Number of Images

This graph 8.6 shows the behavior of accuracy with respect to the parameter k and the number of images used. Unlike the previous curve, we can see that this one is decreasing learning curve.

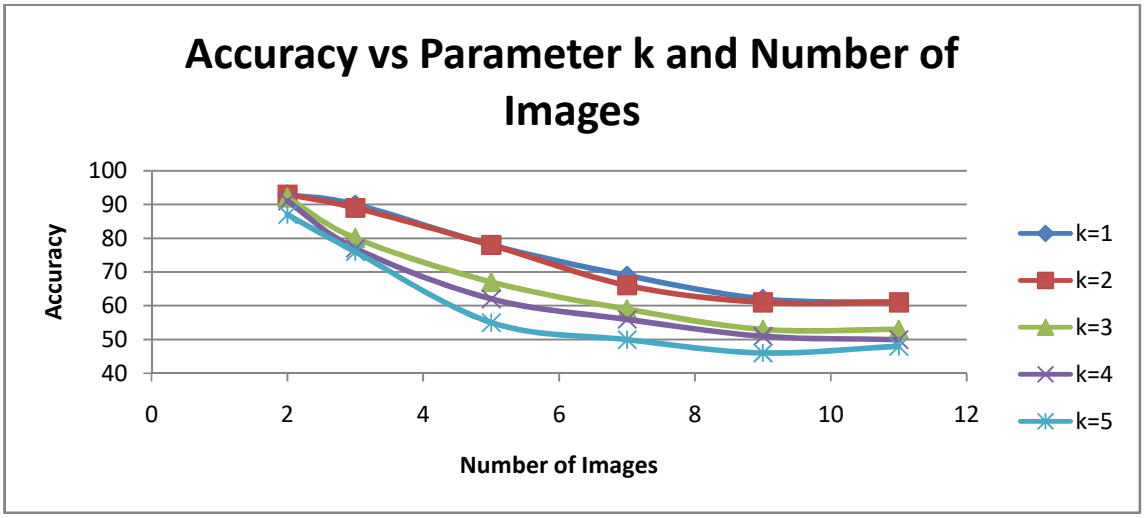


Figure 8.6. Accuracy vs Parameter k and Number of Images Graph

Maximum Memory Consumed and Time Taken vs Number of Images

These graphs 8.7 and 8.8 show the relationship between the maximum memory consumed in GB and the time taken for the same with the number of images. It is a conclusion that memory consumed increases with the number of images and time had a linear relationship with the number of images, which is in correlation with the previous sub-approach.

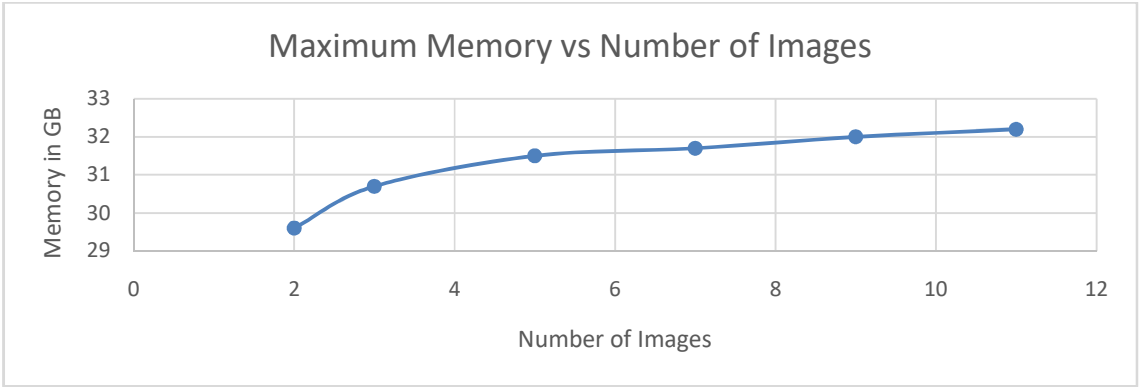


Figure 8.7. Maximum Memory vs Number of Images Graph

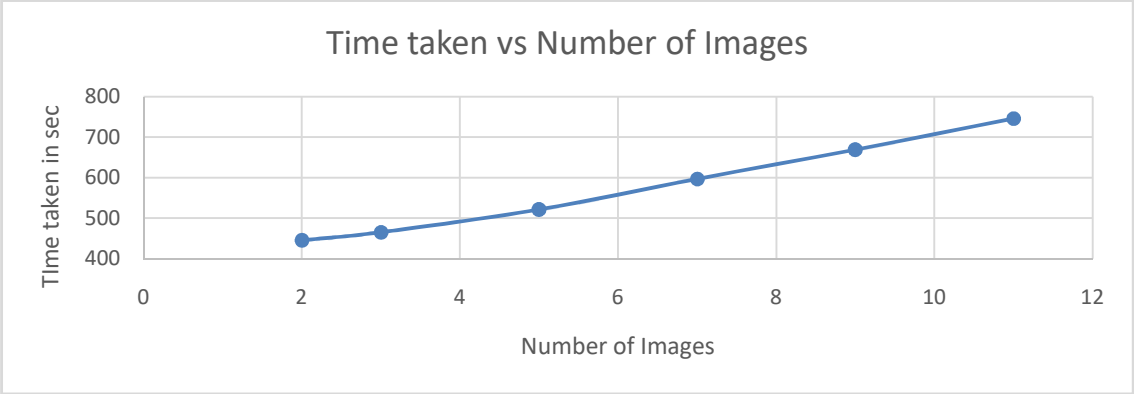


Figure 8.8. Time Taken vs Number of Images Graph

8.3. RESULTS OF INTELLIGENT APPROACH

It was observed that, when we started out with 2 images in the beginning, a decision was taken with next image, thus the number of decision made and the size of the image was tweaked to see the behavior of the system.

8.3.1. Number of Extra Decisions Made vs Size of Image

In this table we are observing the number of times a decision was made after the system was able to arrive with the exiting images. It was further seen that the number of such decisions did not exceed 1.

Table 8.1. Number of Decisions made for Various Sizes of Image

Number of Decisions made	Size of Image
4	32x32
2	100x100
4	200x200
7	300x300
3	400x400

Time taken vs Size of Image

The time taken for computation increases with increase in the size of the image. The behavior is almost linear as seen in the previous curve.

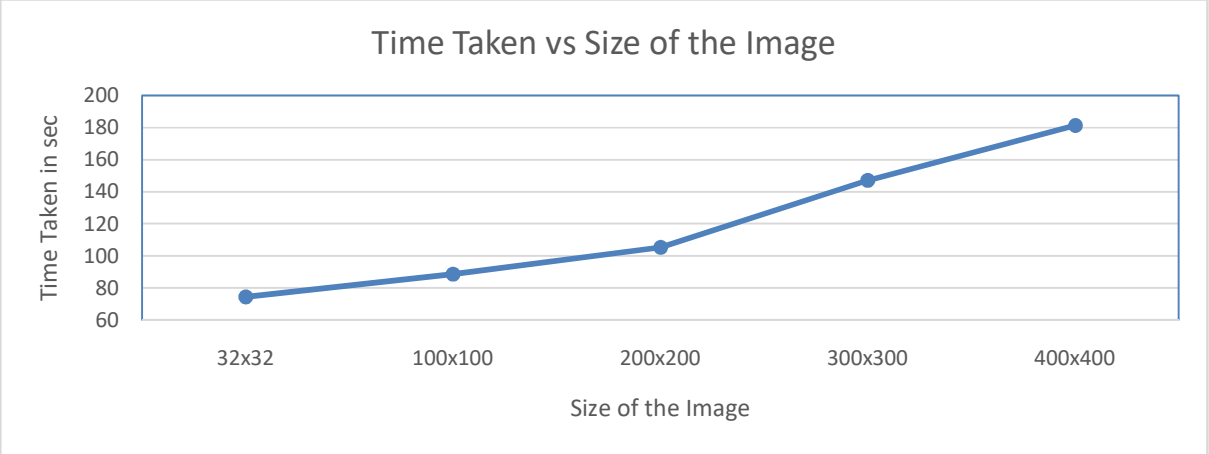


Figure 8.9. Time Taken vs Size of Image Graph

CHAPTER 9

CONCLUSION AND FUTURE WORKS

9.1. CONCLUSION

From inference of the results, it can be concluded that the best method was to go with the Intelligent Techniques as it is the most efficient in terms of implementation as it is the best imitation of the mind of all the approaches.

The optimal size of the image can be 32x32x3 as it on par with the other sizes, but is far more effective in terms of both time taken for execution and memory consumed.

It is also concluded that k=1 is the best value of k, as it works the best amongst all the techniques, i.e. both conventional and intelligent.

9.2. FUTURE SCOPE AND REAL-TIME IMPLEMENTATION

This section focuses on the ideas and thoughts on the implementation of a real-time model that would be feasible on a commercial scale.

9.2.1. Placement of Cameras

The proposed system in this project uses an image-based approach in which the input image is acquired by the cameras. The cameras are positioned in the middle and not the edges as it will be able to get an image with wide field of vision and also, it might be very close to the bus stop, thus resulting in an image that might not be what is expected of. Since the approach focuses on a real-time implementation, the images acquired from this project was captured on camera present in the Moto G4 Plus. The images were captured for the dataset creation was done using burst mode, with a 11.9 MP configuration. This can be seen in the

following image that the cameras would have a certain field of vision in which the images can be captured.

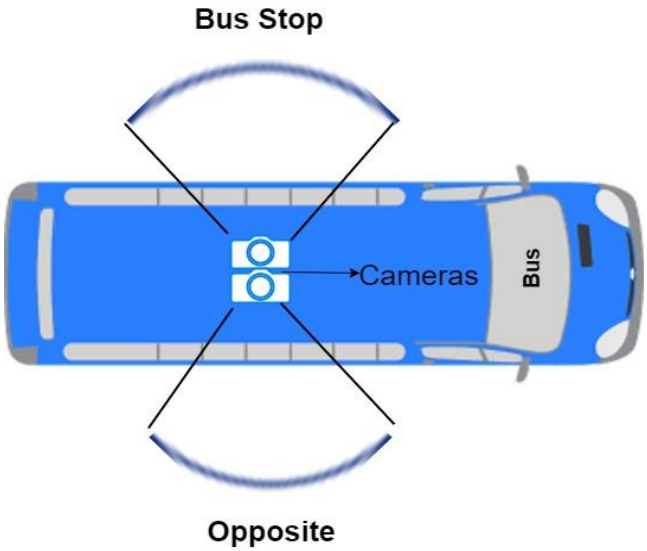


Figure 9.1. Camera Placement in Real-Time

The placement of the camera also correlates with the mimicking of the human mind. As previously discussed, if an individual is asked to identify the bus stop. They intuitively look to their left and right and then conclude on the bus stop. So, a particular decision is made here, the significance of this decision is further discussed in Chapter 7.

9.2.2. Trigger Conditions

Buses usually tend to slow down near the bus stop, in order to stop for passengers to get out and inside the bus. However, the system must be able to announce to the users that the destination has arrived to the passengers, thus this requires us to capture an image just before arriving at the bus stop. For this, a trigger is required. The ideas for implementing this trigger is as follows:

Use of a Speedometer

- In this approach we use a speedometer as a trigger, that is, in case the bus is going below 10 KMPH, as it would prevent the images captured to be blurry thus leading to misclassification, then it is decided that the bus stop has arrived and that the system should capture images of the same.
- However, in such a system, there might a point of failure where the images might be captured for a point of stopping such as a signal where a bus stop might not be present. Thus, the system would announce even if there are no bus stops.
- An advantage of this system is that it requires no human interaction thus, it is considerably independent.

Use of Bus Doors

- In most buses nowadays, there are automatic doors present that, the driver controls them using a button as input. These are invoked when the buses just about reach the bus stop. Hence, the capture of the image can also be tied to the same and thus, we can send it to the recognition engine which would in turn classify the same.
- This system would fail in case, the driver opens the door for the passengers during the commute before reaching the bus stop, despite the intelligent pattern recognition technique which is discussed in the later sections.
- However, an advantage to the system is that, as said above, since there are a lot of buses recently that use automatic doors, the trigger sequence would require the same human intervention as it currently uses and no more further. Thus, even though this system might not be as independent as its predecessor, it does not add any further burden on the bus personnel.

Use of a Separate Button

- Other than previously proposed bus door system, we can implement the trigger functionality separately by the use of a button for the purpose of capturing the image.
- Thus, the images would be acquired from the cameras when the trigger is activated manually by the driver or the conductor, and this image will then be sent to the recognition engine to classify the same.
- But such a system might be adding more jobs for the individual responsible for said task. Even though it requires human intervention, it would not fail assuming the individual has pressed the button when the bus stop arrives and at a low speed, i.e. 10 km.hr, in order to avoid blurry images to be captured.
- It can be concluded that the conductor could handle this duty so as to reduce the burden on the driver.

9.2.3. Recognition Module

The Recognition Module is used to identify the images captured during the previous phase. There is initially a preprocessing module that is discussed in the later chapters. The algorithm chosen for recognition/classification of bus stops in this project is the kth Nearest Neighbors algorithm. Although resorting to use of complex models such as neural networks and such might result in better performance if used along with suitable hardware such as powerful microcontrollers and processors.

9.2.4. Announcement to the Passengers

The last building block of the proposed system is the announcement of the classified bus stop to the passengers in the bus. This system uses the output of the

previous system and intimates the passenger regarding the bus stop. The major approaches to this module are discussed as below.

Use of an LCD display

This is a pretty straightforward approach in which the output of the previous system, which is a class label is displayed to the passengers in the bus. But this approach fails when it comes to dealing with the visually-impaired, yet is a cost effective one. The prototype system implemented in this project uses this approach.

Use of Voice Agent

In this method, a voice agent would announce the classified bus stop to the passengers, thus solving the issues concerning the blind people. Thus, it can be implemented in the existing system itself.

Use of an Application

In this approach, we can use an android application installed in the passengers' phones, thus intimating them every time a bus stop is classified. This would lead to more privacy and obviates the need for a public announcement such as the previous approaches.

9.2.5. Database Update Module

As the environment around the bus stops are subject to change, the database for the train and test data must be updated periodically, preferably every 2 weeks or so. The images can be acquired when the same is tested, as it provides more naturalistic setting to capture images for updating the database.

A test image can be added to the database depending on a successful decision made in the Intelligent Pattern Recognition of the system, or it can also be added based on the time when there are fewer vehicles on the road. For example, probabilistically, most of the roads are devoid of the traffic and other obstructions,

thus the images corresponding to the successful decision made can be added for the updating the database.

9.2.6. Flow of Real-Time System

The following flow chart comprises of all the above components discussed in the design of the system.

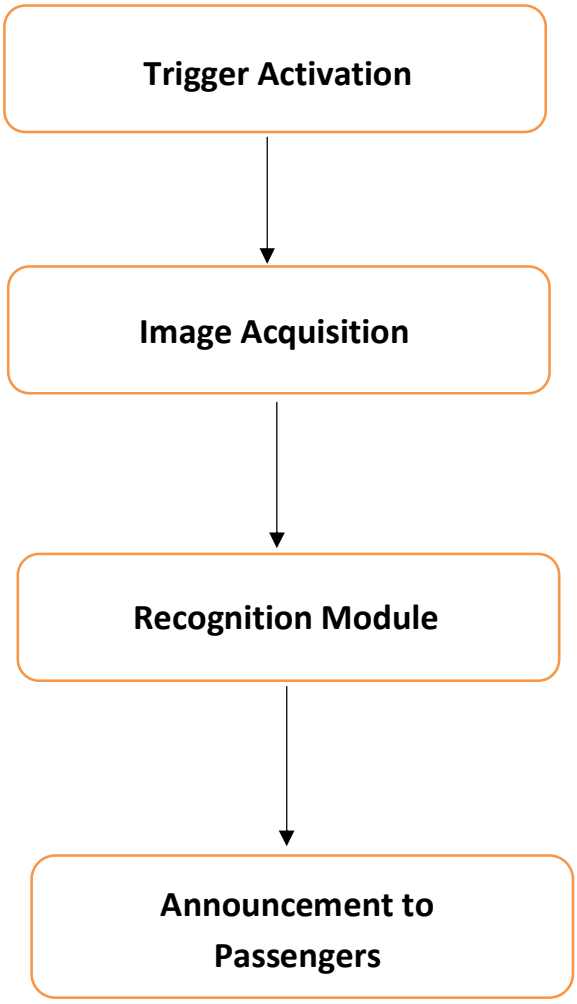


Figure 9.2. Flow Chart of Real-Time System

9.2.7. Overall Flow of Real-Time Implementation

- Upon activation of the trigger present in the system, the images necessary for the classifier are captured.
- The images acquired are preprocessed and further computations for classification of the image are executed in the Recognition Module.
- The output that is recovered from the recognition module is fed into the announcement system, which in turn would intimate the passengers either publicly or privately depending on the announcement system.

REFERENCES

1. Hangrong Pan, Chucai Yi, Yingli Tian, “A Primary Travelling Assistant System of Bus Detection and Recognition for Visually Impaired People”,IEEE International Conference on Multimedia and Expo Workshops (ICMEW), San Jose, CA, 2013, pp. 1-6.

2. Jalila Al Kalbani, Rajaa Bait Suwailam, Arwa Al Yafai, Dawood Al Abri, Medhat Awadalla, “Bus Detection System for Blind People using RFID”, 2015 IEEE 8th GCC Conference & Exhibition, Muscat, 2015, pp. 1-6.

3. U. Franke, A. Ismail, “Recognition of bus stops through computer vision, “IEEE IV 2013 Intelligent Vehicles Symposium. Proceedings” (Cat No. 03TH8683), 2003, pp. 650-655.