Technical Report

ASL DETECTION

Submitted in partial fulfillment of the requirements for the course of

ENCE688D/ENSE698E – Sensor Systems

by

Adheesh Chatterjee (UID –116236935)
Varsha Eranki (UID – 116204569)
M.Eng Robotics

PROJECT SUPERVISOR
Prof Pamela Abshire
Prof David Lovell

A. James Clark
SCHOOL OF ENGINEERING

TABLE OF CONTENTS

ABSTRACT

Verbal communication is the most effective medium of conveying ideas and thoughts, but the same is not usually feasible for people with speaking and hearing impairments. They face many obstacles in effective communication with their peers and people in general and even then, not every person they talk to is trained at sign language which involves simple yet precise hand gestures. In our project, we design, develop and prototype a sensor system that detects the hand gestures and converts them to a desired output of either text or speech. We do this by first developing our own dataset of the hand gestures of the American Sign language (ASL), which we use to train our model using various machine learning algorithms. Then, through a simple camera we capture images which are tested on the pre-trained model and a visual or verbal output of the message is generated as required. This sensor system is bound to make communication more flexible, meaningful and articulate to people with impairments, by restraining their constraints for a better communication.

Keywords: Sensor System, American Sign Language (ASL), Machine Learning, Camera

CHAPTER 1
INTRODUCTION

1.1 Problem Definition

A sensor system is a system that measures a property by detecting and capturing the input, such as touch, wavelength, pressure, temperature, or acceleration, and responding with appropriate feedback.

ASL stands for American Sign Language and is a visual mode of communication
The shape, placement, and movement of the hands, as well as facial expressions and body movements, all play important parts in conveying information.

Communication is the most basic mode of conveying thoughts and ideas, and verbal communication being the most effective remains a constraint for hearing impaired and speech impaired people. In such circumstances, a sensor device system that senses the American Sign language (ASL) as its input and captures the hand gestures, processes it and converts it to a voice output will bridge the gap between them and the rest of the world. Such a system will essentially make their communication more productive and flexible, mainly at workspaces where ideas are shared most often.

At food and beverage chains, where verbal communication is the sole means, hearing and speaking impairment is a major constraint while hiring. Such people face difficulties in understanding their customers, and the environment suddenly becomes less friendly leading to lesser job opportunities. Sometimes, the inability to communicate verbally becomes a constraint at home, where not everyone may understand ASL. Not only the person, but also the people around him/her should also be equipped in ASL.

1.2 Outline

In our project, we design a flexible sensor system that applies a simple Supervised learning aspect of the Machine Learning technique from a labelled training dataset that we predefine. To develop such a sensor system, which uses the hand gestures of American Sign language, we feed the dataset containing these hand gestures using a simple camera.

Various machine learning techniques are employed that implement the application of probability in sorting the testing dataset and comparing it to the training dataset. Once

the testing data is compared with the predefined training dataset, the algorithm classifies the testing data to its required intermediate response.

Then filters are applied to the input data by comparing it with the predefined dataset and filtering the interference and noise aspect, and the capability to detect the hand gestures accurately and classifying it to the appropriate gesture notation. The filtering aspect does not usually play much of a role as the capture frames are programmed to identify the hand by restraining the background in the form of a defined databox. This databox is fixed to the hand gestures and identifies it from the camera.

The hand gestures are matched to the letters with the help of the machine learning algorithm, that picks the training data code to which the test data corresponds to.
After the dataset values are identified and the letter response is generated, the system now interprets it in the form of a text that is detectable easily. Now this letter response is being generated as the training dataset was coded with its respective letter response in the ASL and hence, the output is sent. The texts can also be served as one of the output responses if there is no need for verbal response.

However, if there is a requirement, which in most cases is, the text response of the hand gestures recorded as the input is converted to a verbal response by the implementing of Google Text-to-Speech application, which converts the text format into an audio format. This system is portable and as most of the data is fed into the system and the equipment used are necessarily common, it can be labeled as an effective and easy means of verbal communication for people with speaking and hearing impairment.

1.3 Objective of Work

- To design, develop and prototype a sensor system that detects hand gestures and converts them to speech.
- To use Image Processing to get filtered image and converting into machine readable format
- To use various Machine Learning algorithms for data processing and optimizing to get the best one
- To use Neural Networks for data processing and fine tuning hyper parameters for optimization

1.4 Motivation:

Most of us cannot and do not know how to communicate with people having earing and speaking impairment, both of us hence restrain from coming to a verbal contact. Recently, in October 2018, Starbucks at Washington DC hired people with this impairment so as to address the most general public who face such issues on a daily basis.

Our sensor system works in both ways, for the audience as well as the actors. It makes it user friendly and restores the ability to hear and be able to speak to the general public even without them knowing the gesture notation.
It efficiently converts the hand gestures and gives a voice to the people who have been restrained on many ways. Even though one tries to ignore and neglect the emotional distress that is a setback for few people on daily basis, this sensor system is an effective approach to curb their issues.

1.5 Design elements Included:

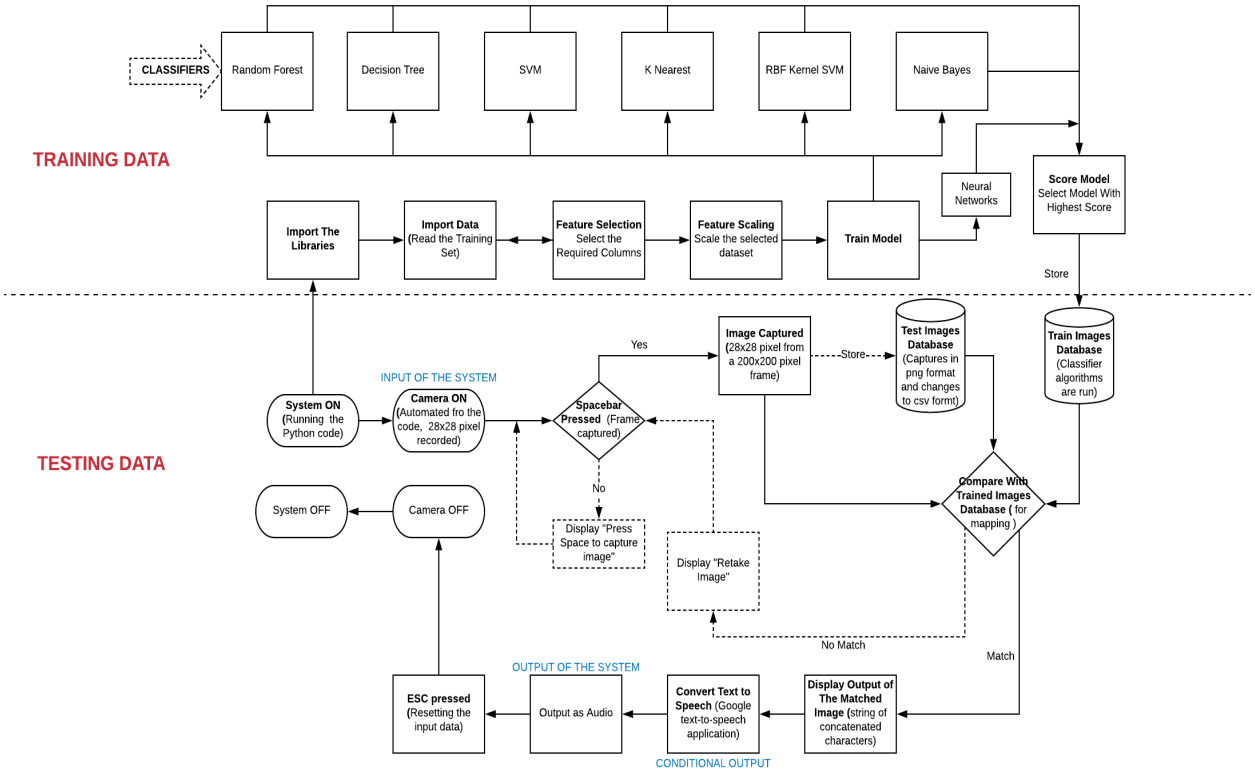
☐ Engineering Standards	☒ Prototype and Fabrication
☐ Design Analysis	☒ Experimentation
☒ Modelling and Simulation	☒ Software Development

1.6 Realistic Constraints to be addressed

☒ Economic	☐ Ethical
☐ Environmental	☐ Health and Safety
☐ Social	☒ Manufacturability
☐ Political	☒ Sustainability
☐ Legal	☐ Regulatory

CHAPTER 2
METHODOLOGY

2.1 Activity Diagram



The above diagram shows the flow of data that our sensor system model.

2.2 Use Case

1. The Relevant Environment: The ASL interpreter system can be deployed in any real environment, but it can find its best use in work environment and conferences where verbal communication is what all matters. Although, sometimes it could be problematic for the present system to detect the hand gestures in bad lighting or too much lighting. It could also be a constrain to detect hand gestures while under dynamic conditions as the capture frames may not be accurate to distinguish the gestures.

2. Actors: The actors in the designed ASL system involves people with hearing and speaking impairment and the general people to whom the sign language is being interpreted. The data controller and micro controller in the system, although termed as entities, aid in performing the system movements and data.

3. Initial values: The initial values for the system to initiate its functioning include the supply of power, camera being turned on with zero detection of hand gesture and a thorough check for accuracy of the better classifier being ready to evaluate the future capture frames.

4. Stimulus: The stimulus in this case is the is the text and audio interpreter which gives the output in the form of letters and words. Sentences may also be an option but may seem time consuming. The stimulus is produced using Google-text-to-speech converter.

5. Events: The events in the system model are clearly the immediate responses after the processing of the png files taken through the capture frames. The event is a continuous format of generating responses with continuous inputs. The program plays on a loop for every capture frame to generate the continuous timed responses.

6. Closure: Although one can terminate the loop with the help of the programmed controller, one can end the system running and exit the model by limiting the duration, setting a limit or even monitoring it intermittently.

7. Sensors: In this system, most processes not only play the role of sensors but also help in processing the signal from one place to another, say, the algorithm for training and testing the dataset acts as a sensor for detecting the values and mapping it to the corresponding value and also as an operative in calculating the input capture frame in order to input it to the classifier.

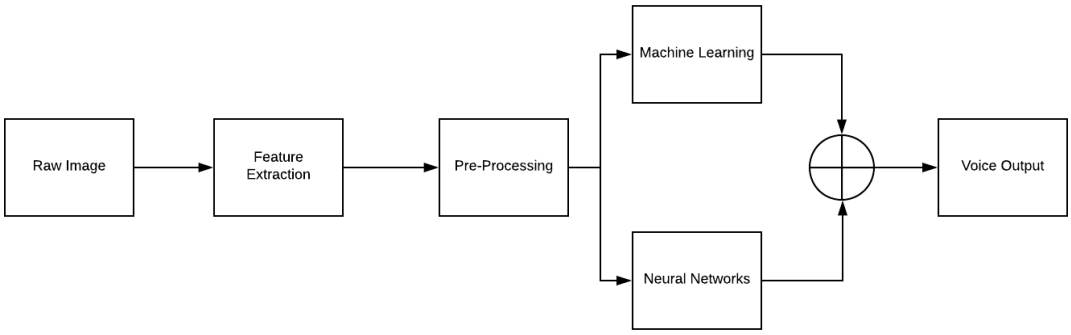
2.3 Application:

This sensor system can be applied throughout where speech is the only medium of communication. In most general cases, operating on hearing and speech impairment medical cases is not effective when it comes to terminating the problem, and there are cases, when the condition is inoperable. That is where this system comes into picture. It can be used in a work environment, homes and any general hospitals by making these places more convenient and people friendly.

2.4 Functionality:

This sensor system is very economical as the components used the components are quite common and easy to build. The camera which plays the role of the sensor, is used for detecting and capturing frames of the hand gestures. We used Python and imported Open CV library, Numpy library, Pandas library and GTTS to code the algorithms involved in building a dataset, training and testing the dataset, classifying the dataset values and the for processing of the input given to the camera to produce the audio output. Algorithms used to classify these dataset values are predefined and use probability as their main criterion. Any Python platform can be used to run the code for operating this sensor system.

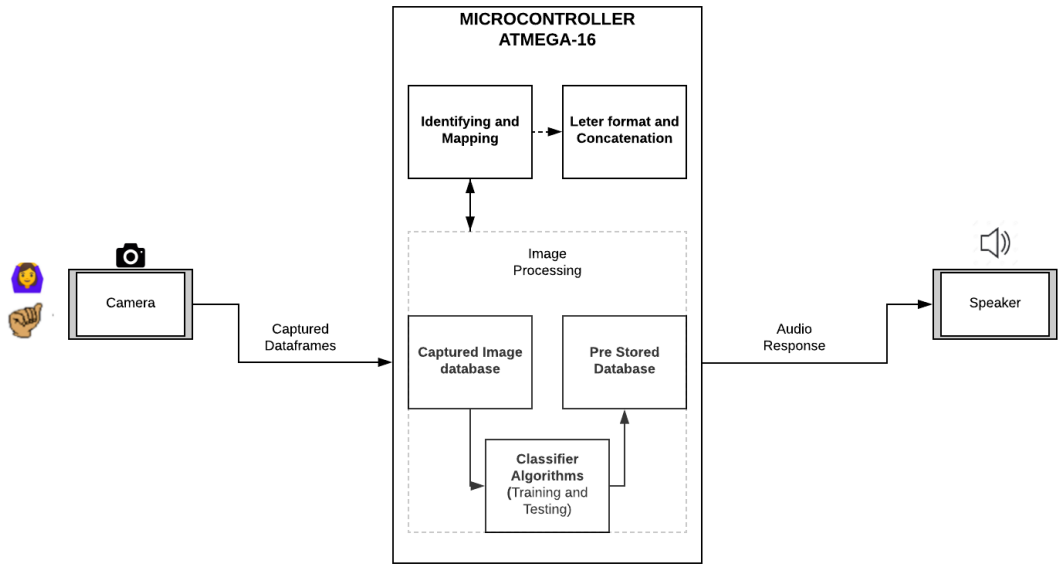
However, we ran the code on a PC and for including the property of portability, the same can be implemented on a pin camera coupled to a microcontroller for running the code and establishing the voice response using Google text-to-speech application.



Moreover, since the data is programmed, it is easy to use different hardware devices by simply replacing the stored data specifications with the new data specification and run the code again. For example, we used a 200x200 pixel data for the dataset with a certain camera, now, this property can be replaced with any other pixel data value by replacing the same in the code.

CHAPTER 3
SENSOR MECHANISM, WORKING AND TECHNOLOGY IMPLEMENTED

3.1 Internal Block Diagram



3.2 Image Processing

RGB – RGB image processing is done by forming a composite of three independent grayscale images that correspond to the intensity of red, green, and blue light. Gathering RGB data is also straightforward because RGB imagery boils down to intensity. Combination of R, G and B gives different colours.

Grayscale - In Grayscale image processing we do not differentiate how much we emit of the different colours, we emit the same amount in each channel. What we can differentiate is the total amount of emitted light for each pixel; little light gives dark pixels and much light is perceived as bright pixels.

3.3 Machine Learning

Machine learning (ML) are algorithms that computers use to progressively improve their performance on a specific task. They are of 3 types –

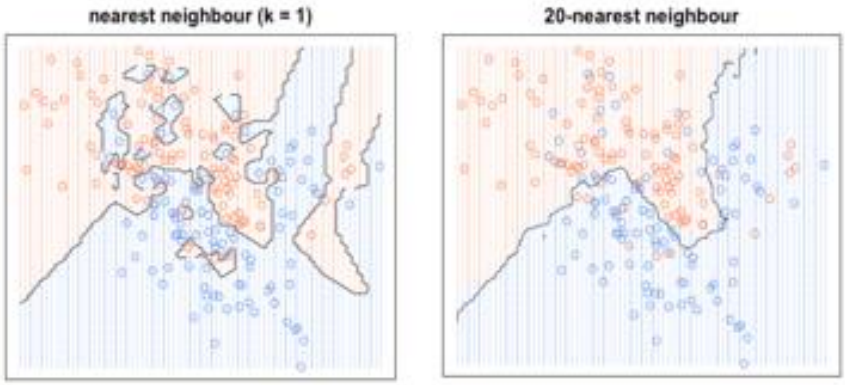
- **Supervised learning** is the machine learning task of learning a function that maps an input to an output based on input-output pairs.
- **Unsupervised learning** is a branch of machine learning that learns from test data that has not been labelled, classified or categorized by identifying commonalities in the data.
- **Reinforcement learning** is an area of machine learning concerned with how agents ought to take actions in an environment so as to maximize some notion of cumulative reward

CHAPTER 4
MATHEMATICAL MODEL

This project doesn't really have a defined mathematical model as its mainly processed through code. However, the algorithms used are explained in detail. The mathematical aspect of the algorithms is highly complex and beyond the scope of this report as they aren't sensor specific.

4.1 KNN

- The K nearest neighbors (KNN) algorithm works based on the assumption that similar things exist in close proximity and hence can be grouped together.
- KNN firstly, loads the data and initializes K to our chosen number of neighbors with an unseen observation x and Euclidean distance d
- It then runs through the whole dataset computing d between x and each training observation. We'll call the K points in the training data that are closest to x the set A, where x is an unseen observation.
- It then estimates the conditional probability between the distance and index value for each class by sorting the A of distances and indices from smallest to largest.
- Finally, our input x gets assigned to the class with the largest probability.



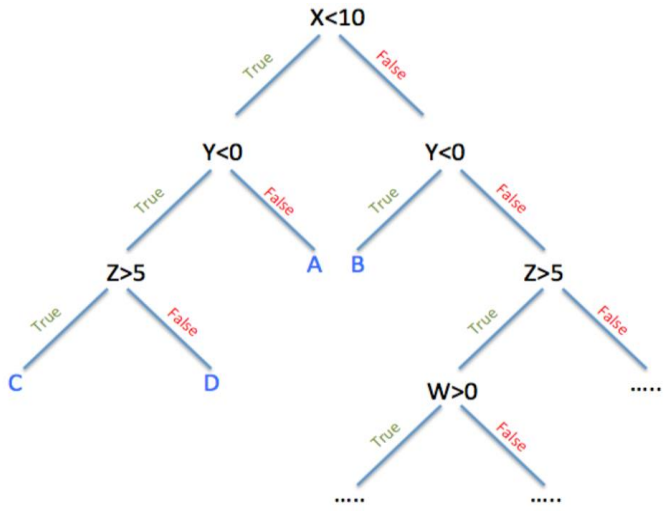
$$P(y = j|X = x) = \frac{1}{K} \sum_{i \in \mathcal{A}} I(y^{(i)} = j)$$

I is the indicator function
which evaluates to 1 or 0

4.2 Decision Tree

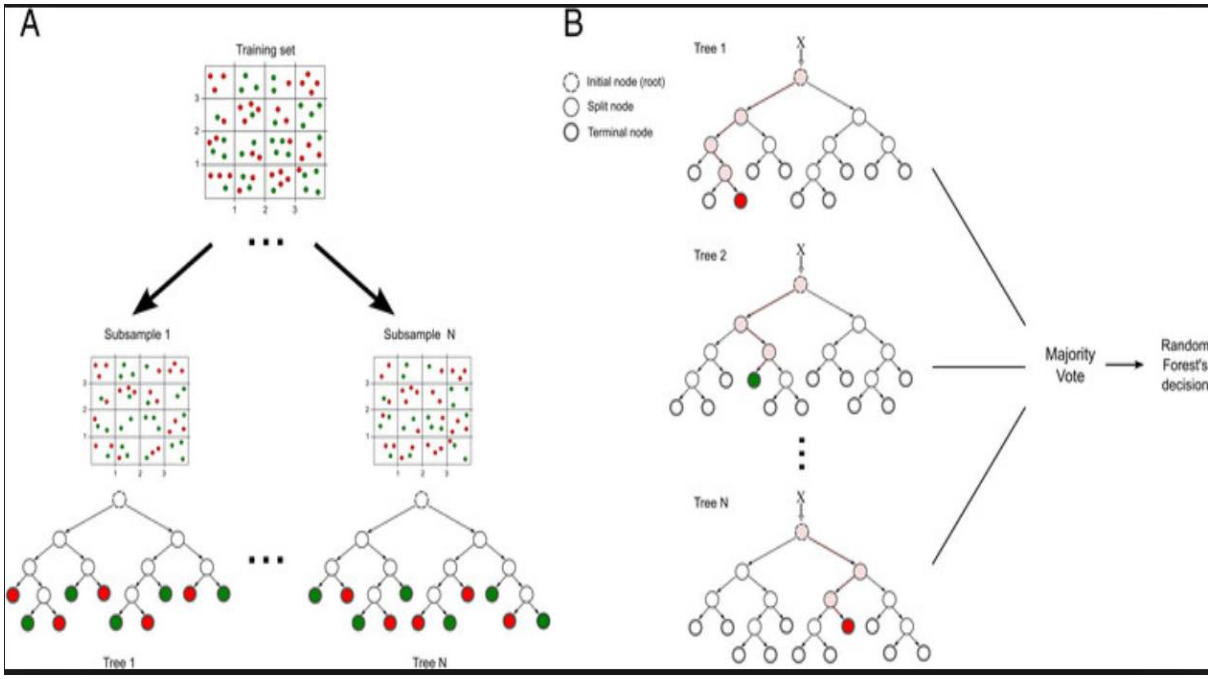
- A decision tree, as the name suggests, uses a tree-like model to make decisions.
- Decision trees learn how to best split the dataset into smaller and smaller subsets to predict the target value.

- The condition is represented as the “leaf” (node) and the decision as “branches” (edges). This splitting process continues until no further gain can be made



4.3 Random Forest

- A random forest is a classifier that groups multiple decision tree classifiers together on various sub-samples of the dataset
- It then uses averaging to improve the control over fitting and getting accuracy for the prediction of the model.



CHAPTER 5
NOISE AND INTERFERENCE MITIGATION

5.1 Sources of Noise and Interference

Image noise is the almost imperceptible specks on a digital photograph taken in good light, from which small amount of information is be derived by sophisticated processing.

Image noises in our context come from bad camera quality, or if the camera is not adaptable for a lighting, or impulse changes in the wavelengths of the EM waves, fluctuations or when subjected to Anisotropic noise, quantization noise and Gaussian noise.

The noises in the sensor system are mainly due to the camera sensor as the input and thus can be classified as follows:

- **Bad Camera Quality** – Although the image being taken is only of the hand , the pixel quality and ISO sensitivity can be responsible for major part of noise in the system
- **Lighting** – Another aspect that comes into play is the the quality of lighting and reflections present that affect the final quality of the image.
- **Salt-and-Pepper Noise** – It is also known as impulse/shot noise. This noise can be caused by sharp and sudden disturbances in the image signal. It presents itself as sparsely occurring white and black pixels
- **Gaussian Noise** – The source of Gaussian noise in digital images is during acquisition of the image. The sensor has this noise due to the level of illumination and its own temperature, and the electronic circuits connected to the sensor provide their share of electronic circuit noise. Bad camera quality causes this noise.
- **Quantization Noise** – This is caused by quantizing the pixels of a sensed image to a number of discrete levels that has an approximately uniform distribution. The conversion of the image to a machine readable dataset causes this noise.
- **Anisotropic noise** – sources show up with a significant orientation in images. The position of the hand plays a major role in determining this noise.
- **Disturbances in region of interest** – Although the region of interest is small, any interference or unrecognized gesture can cause noise.

5.2 Noise Filtering:

- **Mean filter** is an averaging linear filter. The idea of mean filtering is simply to replace each pixel value in an image with the mean value of itself and its neighbours. This has the effect of eliminating pixel values which are unrepresentative of their surroundings.

- **Median filter** is a non- linear filter. The idea of the median filter is to replace each pixel value in an image with the median value of neighbouring entries. This filter is especially good for salt and pepper noise.

- Frequency filter is a type of filter that processes an image in the frequency domain. The image is fourier transformed, multiplied with the filter function and then re-transformed into the spatial domain. Attenuating higher frequencies results in a smoother image and attenuating lower frequencies enhances the edges. The 3 types that can be used are high pass, band pass and low pass

5.3 Noise Reduction Tradeoffs and Assumptions:

While selecting a noise reduction algorithm, we weighed several factors:

- The available computer power and time available: a digital camera must apply noise reduction in a fraction of a second using a tiny onboard CPU, while a desktop computer has much more power and time.
- Whether sacrificing some real detail is acceptable if it allows more noise to be removed (how aggressively we can decide whether variations in the image are noise or not)
- The characteristics involved in the noise and the detail in the image and to better make those decisions.

5.4 Filters that our model can use:

The median is a more robust average than the mean and so a single very unrepresentative pixel in a neighbourhood will not affect the median value significantly. Since the median value must actually be the value of one of the pixels in the neighbourhood, the median filter does not create new unrealistic pixel values when the filter straddles an edge. For this reason, the median filter is much better at preserving sharper edges than the mean filter.

- **Linear smoothing filters:** One method we used to remove noise is by convolving the original image with some mask that makes up for a smoothing operation or low pass filters. For example, consider the Gaussian mask which comprises elements determined by the Gaussian function. This convolution brings the value of each pixel into closer harmony with values of its neighbors. In general, a smoothing filter sets each pixel to the average value, or a weighted average, of itself and the neighbors; the Gaussian filter is just one possible set of weights. Smoothing filters often tend to blur images, thereby the pixel intensity values that are significantly higher or lower than the neighborhood happened to smudge across the area. But since, we are only interested in well-defined edges and not the over-smoothened images, this filter can be used to enhance intensity of the image.
- **Anisotropic diffusion**: Another method for removing noise is to evolve the image under a smoothing partial differential equation similar to the heat equation, which is called anisotropic diffusion. With a spatially constant diffusion coefficient, this is equivalent to the heat equation or linear Gaussian filtering, but with a diffusion coefficient designed to detect edges, the noise can be removed without blurring the edges of the image.
- **Non-local means**: Another approach for removing noise is based on non-local averaging of all the pixels in an image. In particular, the amount of weighting for a

pixel is based on the degree of similarity between a small patch centered on that pixel and the small patch centered on the pixel being de-noised.

- **Nonlinear filters**: A median filter is an example of a non-linear filter and, if properly designed, is good at preserving image detail. To run a median filter:

  1. We consider each pixel in the image
  2. Sort the neighboring pixels in order based upon their intensities
  3. Replace the original value of the pixel with its median value from the list

  Median and other RCRS filters are good at removing salt and pepper noise from an image, and causes relatively little blurring of edges, and hence can be efficiently used in our sensor system.

CHAPTER 6
HARDWARE AND SIMULATION

The Hardware and Simulation implemented are shown in steps below.

Step 1 – Step 8 is for training the program using Machine Learning and Neural Networks.

Step 9 – Step is to test the image for a random input by a user.

Step 1 – The libraries are imported

```
In [1]: import numpy as np
        import matplotlib.pyplot as myplot
        import pandas as pd
        import seaborn as sns
        import pickle
        import os
        import cv2
        from gtts import gTTS
```

Step 2 – Importing the Data and Preprocessing

```
In [2]: train=pd.read_csv('train.csv')
        test=pd.read_csv('test.csv')
```

```
In [3]: train.shape
Out[3]: (27455, 785)
```

```
In [4]: labels = train['label'].values
```

```
In [5]: unique_val = np.array(labels)
        np.unique(unique_val)
Out[5]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8, 10, 11, 12, 13, 14, 15, 16, 17,
               18, 19, 20, 21, 22, 23, 24], dtype=int64)
```

```
In [6]: myplot.figure(figsize = (18,8))
        sns.countplot(x =labels)
```

Step 3 – Implementing Decision Tree

Decision Tree

```
In [13]: #Fitting Decision Tree Classification from the Training Set
         from sklearn.tree import DecisionTreeClassifier
         classifier1=DecisionTreeClassifier(criterion='entropy',splitter='best',random_state=0)
         classifier1.fit(x_train,y_train)
Out[13]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=0,
                     splitter='best')
```

```
In [14]: filename = 'decision_tree.sav'
         pickle.dump(classifier1, open(filename, 'wb'))
```

```
In [15]: #Predicting the Test Set Results
         y_pred1=classifier1.predict(x_test)
```

```
In [17]: #Checking the Accuracy Scores
         from sklearn.metrics import accuracy_score as ac
         acc1=ac(y_test,y_pred1)
         acc1
Out[17]: 0.8872162194973898
```

Step 4 – Implementing KNN

KNN

In [18]:
```
# Fitting KNN classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier2= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
classifier2.fit(x_train,y_train)
```

Out[18]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

In [19]:
```
#Predicting the Test Set Results
y_pred2=classifier2.predict(x_test)
```

In [20]:
```
#Checking the Accuracy Scores
from sklearn.metrics import accuracy_score as ac
acc2=ac(y_test,y_pred2)
acc2
```

Out[20]:
```
0.9928371980089838
```

Step 5 – Implementing Random Forest

Random Forest

In [21]:
```
#Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier4 = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier4.fit(x_train, y_train)
```

Out[21]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [22]:
```
#Predicting the Test Set Results
y_pred4=classifier4.predict(x_test)
```

In [23]:
```
#Checking the Accuracy Scores
from sklearn.metrics import accuracy_score as ac
acc4=ac(y_test,y_pred4)
acc4
```

Out[23]:
```
0.867427461454413
```

Step 6 – Implementing a Neural Network

CNN

In [30]:
```
model = Sequential()
model.add(Conv2D(64, kernel_size=(3,3), activation = 'relu', input_shape=(28, 28 ,1) ))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(64, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(64, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.20))
model.add(Dense(num_classes, activation = 'softmax'))
```

In [31]:
```
model.compile(loss = keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
```

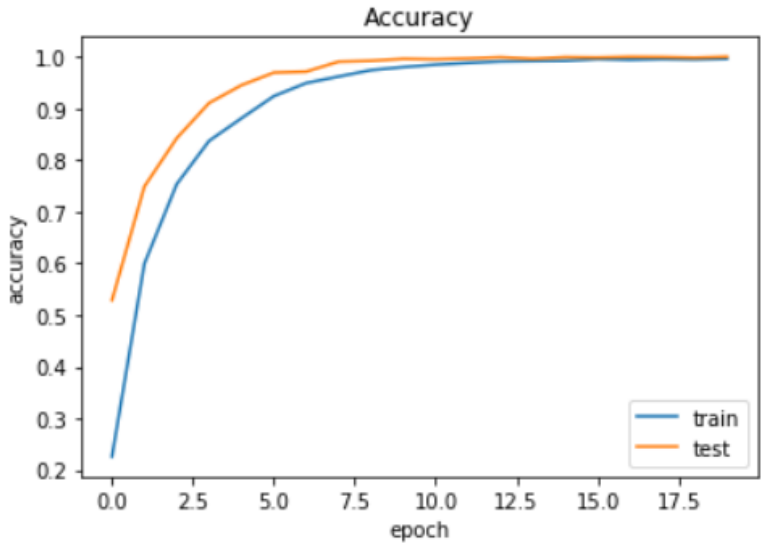
Step 7 – Training the Neural Network

```
In [32]: history = model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=epochs, batch_size=batch_size)
```

```
Train on 19218 samples, validate on 8237 samples
Epoch 1/20
19218/19218 [==============================] - 38s 2ms/step - loss: 2.5157 - acc: 0.2263 - val_loss: 1.4911 - val_acc: 0.5288
Epoch 2/20
19218/19218 [==============================] - 44s 2ms/step - loss: 1.1949 - acc: 0.5993 - val_loss: 0.7760 - val_acc: 0.7487
Epoch 3/20
19218/19218 [==============================] - 33s 2ms/step - loss: 0.7289 - acc: 0.7534 - val_loss: 0.4848 - val_acc: 0.8423
Epoch 4/20
19218/19218 [==============================] - 32s 2ms/step - loss: 0.4794 - acc: 0.8371 - val_loss: 0.3038 - val_acc: 0.9104
3
Epoch 5/20
19218/19218 [==============================] - 37s 2ms/step - loss: 0.3567 - acc: 0.8809 - val_loss: 0.2058 - val_acc: 0.9449
Epoch 6/20
19218/19218 [==============================] - 38s 2ms/step - loss: 0.2403 - acc: 0.9237 - val_loss: 0.1242 - val_acc: 0.9692
Epoch 7/20
19218/19218 [==============================] - 43s 2ms/step - loss: 0.1661 - acc: 0.9490 - val_loss: 0.1097 - val_acc: 0.9711
Epoch 8/20
19218/19218 [==============================] - 41s 2ms/step - loss: 0.1264 - acc: 0.9616 - val_loss: 0.0510 - val_acc: 0.9904
Epoch 9/20
```

Step 8 – Testing the Neural Net

```
In [33]: myplot.plot(history.history['acc'])
         myplot.plot(history.history['val_acc'])
         myplot.title("Accuracy")
         myplot.xlabel('epoch')
         myplot.ylabel('accuracy')
         myplot.legend(['train','test'])
         myplot.show()
```




```
In [35]: score=model.evaluate(x_test,y_test)
         score
```

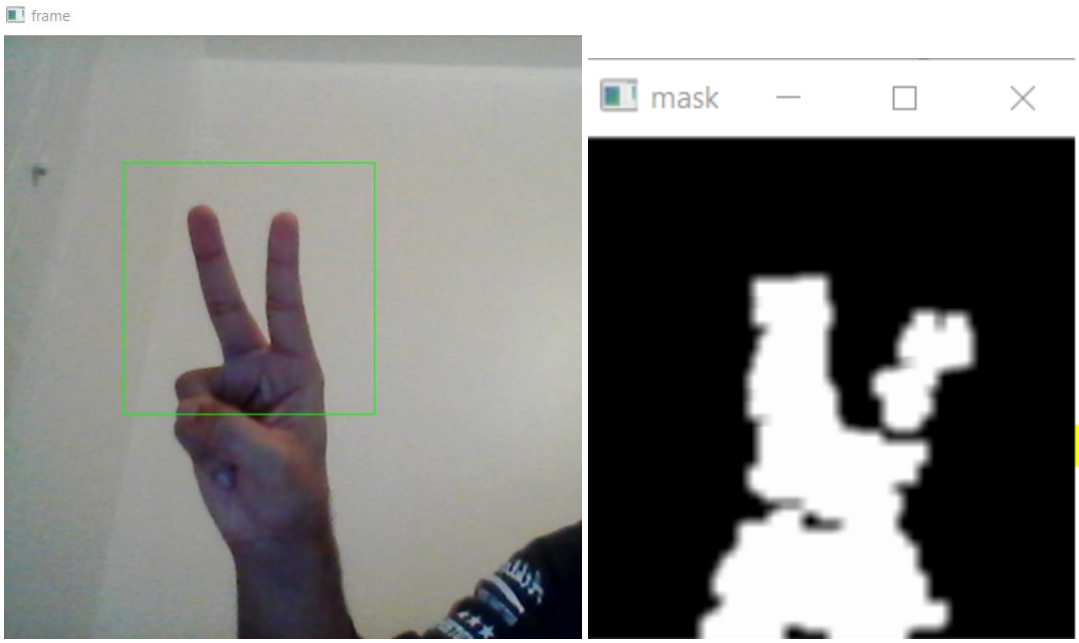
```
8237/8237 [==============================] - 5s 585us/step
```

```
Out[35]: [0.0019948464445945038, 0.9998785965764234]
```

Step 9 – Capturing Images

```
cam = cv2.VideoCapture(0)
img_counter = 0

while True:
    ret, frame = cam.read()
    frame=cv2.flip(frame,1)
    kernel = np.ones((3,3),np.uint8)
    roi=frame[100:300, 100:300]
    cv2.rectangle(frame,(100,100),(300,300),(0,255,0),0)

    hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

    # define range of skin color in HSV
    lower_skin = np.array([0,20,70], dtype=np.uint8)
    upper_skin = np.array([20,255,255], dtype=np.uint8)

    #extract skin colour image
    mask = cv2.inRange(hsv, lower_skin, upper_skin)

    #extrapolate the hand to fill dark spots within
    mask = cv2.dilate(mask,kernel,iterations = 4)

    #blur the image
    mask = cv2.GaussianBlur(mask,(5,5),100)

    #Display the windows
    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)

    if not ret:
        break
    k = cv2.waitKey(1)

    if k%256 == 27:   # ESC pressed
        print("Escape hit, closing...")
        break
    elif k%256 == 32: # SPACE pressed
        img_name = "{}.png".format(img_counter)
        cv2.imwrite(img_name, roi)
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()
```

On running the above code, we open the camera



In the left image is what the user sees and the right is what the machine sees. This is then converted from a png file to a csv file for processing

Step 10 – Convert .png to .csv

CONVERT .PNG to .CSV

```
In [ ]: img_file = '0.png'
        gray_img = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)
        print ('Gray shape:', gray_img.shape)
        np.savetxt("0.csv",gray_img,delimiter=",")
        df=pd.DataFrame(gray_img)
        df.to_csv("0.csv")
```

Result –
Machine learning is based on the idea of dividing our data set into two subsets:
Training set: A subset to train a sensor system model.

Test set: A subset to test the trained sensor system model.
While considering the aspect of choosing the dataset, we made the assumptions that it is large enough to yield statistically meaningful results and is representative of the data set as a whole.

Assuming, that our test set meets the preceding two conditions, our goal was to create a model that generalizes well to new data. Our test set serves as a proxy for new data. For example, take into notice that the model learned for the training data is very simple. This model does not do a perfect job, say, few predictions are wrong. However, the model does about as well on the test data as it does on the training data. In other words, this simple model does not overfit the training data.

CHAPTER 7
TRADEOFFS AND CONSTRAINTS

7.1 Tradeoffs

The system although being portable and economical, does face few tradeoffs when it comes to characteristics such as accuracy, time constraint, gesture constraint and speed of response. This data can however be modulated overtime and various other applications of algorithms will fix these issues.

When it comes to accuracy, the hand gestures clearly involve letters of alphabet and not words, which again should be converted to word format in our case. Essentially, even though the vocabulary by these hand gestures can construct any word, the hand gestures used for words physically, can be interpreted faster when compared to hand gestures used for letters.

The hand gestures for have a huge time constraint and may seem less productive for emergency situations where data is to be processed to build words at the rate of seconds. The operation time is slower even though the computation and processing time is faster.
Speed of response is the main drawback of this system that is being compromised at great level. Although it magnifies the aspect of being user friendly, one may find it tiresome to use it for a longer time.

However, we choose to further develop the system by automating most of the criterion involved and implementing word gestures as well to reduce the complexity in executing it by the people.

7.2 Constraints Involved and Optimizing:

The system although being portable and economical, has drawbacks when it comes to characteristics such as accuracy, time constraint, gesture constraint and speed of response. When it comes to accuracy, the hand gestures clearly involve letters of the alphabet and not words.

- Essentially, even though these hand gestures can construct any word, the hand gestures used for words physically, can be interpreted faster when compared to hand gestures used for letters.
- The hand gestures have a huge time constraint and may seem less productive for emergency situations where data is to be processed to build words at the rate of seconds. The operation time is slower even though the computation and processing time is faster.
- Speed of response is the main drawback of this system. Although it magnifies the aspect of being user friendly, one may find it tiresome to use it for a longer time. However, we can choose to further develop the system by automating most of the criterion involved and implementing word gestures as well to reduce the complexity in executing it by the people.

Therefore, the sensor system model can be optimised in the following two ways:

1. One way to build an optimal sensor system in this context is by making the system dynamic in real time and generating the outputs with minimal error in speed of response. By implementing this, the video input is processed continuously over time and not only letters, but also words will be generated efficiently. Cameras such as Adafruit Llc or Canary can be used for detecting these hand gestures for a better speed of response as the capture the data frames at the rate of milliseconds and process it.

2. Another way to optimize the sensor system model is by implementing Neural Networks. In our image classification problem, our goal will be to tell which class the input image belongs to. The way we are going to achieve it is by training an artificial neural network on few thousand images of all letters and make the NN (Neural Network) learn to predict which class the image belongs to, next time it sees an image having specific letter in it.

    - The key thing to understand while following this is, that the model we are building now can be trained on any type of class we want. Coming to the coding part, we are going to use Keras deep learning library in python to build our CNN (Convolutional Neural Network). the processes of building a Convolutional Neural Network always involves four major steps: Convolution, Pooling, Flattening and Full connection.
    - The test image holds the image that needs to be tested on the CNN. Once we have the test image, we will prepare the image to be sent into the model by converting its resolution to a specific value as the model only excepts a defined resolution. Then we are using predict() method on our classifier object to get the prediction. As the prediction will be in a binary form, we will be receiving either a 1 or 0, which will represent the right and wrong letter respectively.

CONSTRAINTS / IMPROVEMENTS	Cost Factor	Speed of response (training the data)	Efficiency of the system	Reliability
1. Cameras:				
a. Adafruit Llc	**	****	***	**
b. Canary	****	****	****	***
c. Web cameras (Logitech)	*	**	**	*
d. Multiple cameras	**	*	***	***
2. RBG and Gray Scale	-	-	**	-
3. HSV Scale Image processing	-	-	**	-
4. Classifier Algorithm:				
a. KNN	-	***	99.20%	-
b. Decision Tree	-	*	88.72%	-
c. Random Forest	-	**	86.74%	-
5. Convoluted Neural networks (CNN)	-	*	99.65%	-
6. Noise Interference Filters:	*	-	***	**

CHAPTER 8
VALIDATION AND VERIFICATION

Since we use the concept of Machine Learning and Neural Networks, they optimize and validate themselves as they are both iterative processes.

The validation is done by running the test data through the algorithm, and the accuracy for each algorithm is calculated successfully.

8.1 Validation

The sensor system model meets the demands of the response expected. As the hand gestures are being converted to the text interpretation and then followed by speech interpretation, this effectively models the data set values to the corresponding letter response and the output is timely generated.

Although, the system interpretation can only be precisely identified and judged by the actor giving out the hand gestures, testing can be done to terminate any observable errors. But in real case scenarios, if an impaired person is producing gestures to be converted to speech, errors in interpreting the data can only be identified by that person solely, and not the audience to whom the message is being communicated.

Hence, we overcame this by making our model more accurate by including more varied datasets, training it under various classifier algorithms as mentioned above. The system model can also be validated by testing the model under dynamic conditions, when the capture frames are in movement, which we would develop it in the future. Speed of response of the system is also one attribute we wish to further develop it.

8.2 Verification:

The system was essentially verified under the given conditions of the camera specifications for the pixel data being processed to the dataset values given in the training and testing in the csv file format and the mapping the data to the corresponding letter.

CHAPTER 9
FUTURE SCOPE

The single hand gesture recognition system works successfully for real-time static hand gesture recognition gesture recognition rate and is suitable for ASL alphabets and meaningful words and sentences. But again, if you pay close attention, the gesture detection is being made statically and captures frames in the fixed databox.

The present sensor system of ASL detection and interpretation uses Machine learning techniques to provide a reasonable accurate result depending on the number of datasets being trained and tested. But, by the application of Neural networks, one can build a better accurate sensor system model which would be an intended future development we wish to progress towards.

Since the present system model works only on letters, there is a scope of building a system model that inputs the complete hand and body gestures of the sign languages and inputs them in the form words and processes accordingly. Such systems are a boon to the future of medical technology as well.

REFERENCES

[1] Jerome M. Allen, Pierre K. Asselin, Richard Foulds, "American Sign Language Finger Spelling Recognition System", 0-7803-7767-2/03/\$17. 00 02003 IEEE, March, 2003.

[2] Richard A. Tenant, Marianne Gluszak Brown, "The American Sign Language Hand shape Dictionary", Gallaudet University Press, page no.9, Publication Date: June 1, 1998.

[3] Jinda-apiraksa A, Pongsteinsak W and Kondo T, " A Simple Shape-Based Approach to Hand Gesture Recognition", International Conference on Electrical Engineering/Electronic Computer Telecommunications and Information Technology 2010.

[4] Sudeep D, Gandhali K and Priti K, " Sign Language Recognition using Color Means of Gradient slope Magnitude Edge Images", International Conference on Intelligence systems and Signal Processing 2013

[5] Kunal Kadam ; Rucha Ganu ; Ankita Bhosekar ; S.D. Joshi , “American Sign language Interpreter”, International Conference on Technology for Education, IEEE, 2012.