Application of Machine Learning Techniques for Real-Time Sign Language Detection using Wearable Sensors

Nazmus Saquib and Ashikur Rahman, Member, IEEE

ABSTRACT

Sign language is a method of communication primarily used by the hearing impaired and mute persons. In this method, a letter is expressed by hand gestures and meaningful words are constructed by signaling multiple letters in a sequence which is known as *fingerspelling*. In this paper, a system has been developed to detect fingerspelling in American Sign Language (ASL) and Bengali Sign Language (BdSL) using (data) gloves containing a number of suitably positioned sensors. While designing the system, at first we identify all suitable locations of the sensors in order to properly construct the glove that can detect both the languages. The methodologies employed have the ability to be used even in resource-constrained environments. The system is capable of accurately detecting both static and dynamic symbols in the alphabets. The system shows a promising accuracy of (upto) 96%. Furthermore, this work presents a novel approach to perform a continuous assessment of symbols from a stream of run-time data.

KEYWORDS

Sign Language Recognition (SLR), data glove.

ACM Reference Format:

Nazmus Saquib and Ashikur Rahman, Member, IEEE. 2020. Application of Machine Learning Techniques for Real-Time Sign Language Detection using Wearable Sensors. In *Proceedings of MMSys '20:The 11th edition of ACM's premier conference on multimedia systems (MMSys '20).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Sign language is a method of communication which primarily uses articulation of the hands to convey meaningful messages. It involves a simultaneous combination of hand shapes, orientation and movement of the hands, arms or body, and facial expressions. It is the most common mode of communication among the hearing and speech impaired persons. In other words, sign language is mostly used by the people with speech difficulties due to some sorts of disability.

N. Saquib is with the Department of Computer Science, University of California, Santa Barbara, USA (email: nazmus@ucsb.edu).
A. Rahman is with the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh (email: ashikur@cse.buet.ac.bd).

MMSys '20, June 08–11, 2020, Istanbul, Turkey

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

According to World Health Organization (WHO), over 5% of the world population is hearing impaired [7]. This considerably large community, along with the speech impaired community, uses sign language as the primary mode of communication. When two persons knowledgeable with sign language want to communicate with each other, they can do it conveniently. Problem arises when a normal person without domain knowledge of sign languages wants to communicate with a disabled person. Generally, in this type of scenario an *interpreter* is used. Interpreters translate sign language to oral language and vice versa. However, an interpreter might not always be available. Secondly, use of interpreters could be very costly. A professional ASL interpreter certified by the Registry of Interpreters for the Deaf (RID) has an hourly rate of 30-35 USD [4]. Therefore, it is desirable to have an intermediary system which will translate sign language to oral language and vice versa. This work attempts to partially bridge this gap (at least) in one direction – translating sign language to text.

Automatic recognition of the sign language is a very challenging task because it varies from language to language as the alphabet is different for different languages. For example, Fig. 1 and Fig. 2 shows the alphabet of American Sign Language (ASL) and Bengali Sign Language (BdSL) respectively. It is easy to see that ASL widely varies from BdSL. Sign languages even vary from region to region. For instance, ASL is different from British Sign Language (BSL) although they share the same alphabet. Moreover, some sign languages have both single-handed and double-handed version of certain symbols. Again, sign languages can have a compressed form where a single word is represented by a single gesture. Sometimes instead of using a single gesture for a word sign language allows *fingerspelling* which is a process of spelling out the word by using signs that correspond to the letters of the word. While using sign language, a user might choose for using a sign for the word if available, or the user might choose fingerspelling of the word. There are many words which have not been standardized in the respective sign language dictionaries. According to [5], there are more than 150,000 words in spoken English that do not have ASL counterpart. Moreover, people's names, places, titles, brands, etc., also do not have any standardized symbols. Besides, a user might not know the exact sign for a particular word. In these scenarios, the user must resolve to fingerspelling the required words.

Although traditionally sign language is considered to be an *unaided* method of communication (i.e. relying on only the user's body to convey a message), a substantial amount of research work has been performed to aid sign language recognition using specialized tools. Traditional sign language recognition systems can be broadly classified into two categories: (i) digital image processing based systems and (ii) data glove (a glove with various sensors attached to it) based systems [9]. In the former approach, a large corpus of images of various signs is collected which are then used to train a model using supervised learning. The latter approach

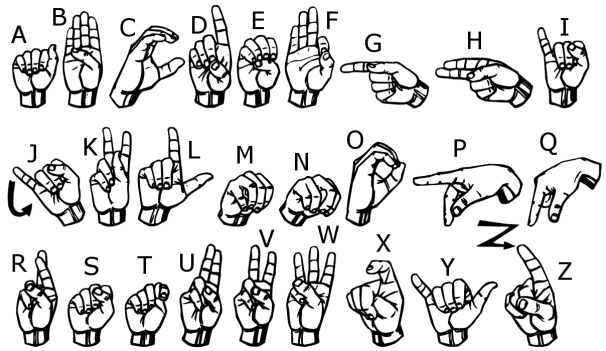


Figure 1: ASL alphabet [1].

follows a similar trend but instead of using images it deploys various sensors that generate necessary signals to train a classifier. In this paper, we focus on one-handed fingerspelling of ASL and BdSL using data glove approach. The major contributions of the paper are summarized below:

- We design and implement a data glove capable of recognizing fingerspelling in ASL and BdSL. Although commercial data gloves are extant, they might not always be readily available to the end-user. Moreover, some of the commercial gloves might need tweaking for specific languages. This work elaborately describes the construction of a data glove using sensors which are much more common than commercial gloves. The design of the glove can be changed to fit a particular language.
- We build a repository system containing a data set of sensor values using the designed data glove. A crucial prerequisite in training a supervised classifier is the availability of labeled data. However, sign language data set of ASL alphabet are not publicly available, and that of BdSL alphabet does not exist at all. This work attempts to ameliorate this situation by generating a comprehensive dataset.
- We propose a system capable of recognizing sign language even under *resource-constrained* environment. Apart from the real-time assessment of fingerspelling, this system provides support during data collection phase.
- Finally, we provide sign language models using data gloves. Although a considerable amount of research is found in the literature on sign language recognition using data glove, most of them provide little insight to *sign language modeling*, that is, determining intelligible messages from a stream of sensor data. This work proposes a methodology to infer characters and subsequently words from a continuous stream of data.

The rest of this paper is organized as follows. Section 2 presents related research works. Section 3 discusses glove construction, data collection, and the process of training a suitable classifier. Section 4 gives an overview of the system and discusses some critical design issues. Section 5 presents experimental results and the findings derived from these results. Finally, Section 6 concludes the paper with some possible directions to future works.

Figure 2: BdSL alphabet [8].

2 BACKGROUND

Over the years, sign language recognition (SLR) has been attempted in two different approaches: (i) digital image processing (DIP) based approach and (ii) data glove based approach. In both approaches, the basic idea remains the same – collecting a considerable amount of data and applying a supervised learning algorithm to classify that data. In digital image processing based systems, the data is simply pictures of various signs. On the other hand, in data glove based approach, the data is just the value of various sensors attached to the glove. Despite the basic idea being the same, there are certain differences between the two approaches that we describe next.

2.1 DIP based Approach for SLR

In DIP based approach, at first, a large set of images is collected. The collection process can be facilitated by either a generic camera or specialized one such as a consumer depth camera [13]. Once the data is collected, various machine learning algorithms are applied to train a model capable of classifying images of different signs. One major drawback of this approach is the data collection procedure is inherently cumbersome, and the quality of the image might be compromised depending on the ambient light of the environment [18]. Another glaring disadvantage is it is not feasible to create a portable system using this vision based approach. Assuming somehow the user can carry the camera with him/her, he/she still has to remain within the field of vision of the camera, thus

restricting the user's distance and motion. In accordance with the popularity of Microsoft Kinect, Dong et al. [10] attempted to use this low-cost depth camera to recognize 24 letters of ASL by using a random forest ML algorithm. Nabiyev et al. [16] applied neural network on images of vowel characters of the Turkish language. Pandey et al. [19] proposed a methodology to segment out the hand region from the image and successively generating a feature vector of ten features. Finally, they applied backpropagation neural network for classification. Liwicki et al. [14] worked on a dataset of 1,000 low-quality webcam videos of 100 fingerspelt words of BSL. They employed Hidden Markov Model (HMM) to interpret probable words from a stream of characters. Naoum-Sawaya et al. [17] carried out various steps to extract suitable templates from raw images, such as histogram equalization, background rejection, skin color extraction and thresholding, morphological filtering, and flood filling. After the successful generation of templates, template matching algorithm was used to classify images. Rekha et al. [21] employed K-Nearest Neighbor (KNN) and Support Vector Machine (SVM) for hybrid classification of a single signed letter. Furthermore, they proposed a lexicon-based approach to recognize fingerspelt words using HMM. Transcending the use of a simple RGB camera, Kuznetsova et al. [13] worked on real-time SLR using a consumer depth camera. The images collected from this camera were used to derive rotation, translation, and scale invariant features. They trained a multi-layered random forest (MLRF) to classify the feature vectors.

2.2 Data Glove based Approach for SLR

A data glove has a number of sensors attached to it to determine various hand movement features such as the bending of fingers, the orientation of the hand, rotational motion of the hand, contact between two fingers, etc. Although some of the data gloves [2, 3] are commercially available, they are not suitable for sign language recognition due to their high cost or unavailability of specific sensors. A glove designed and created from scratch results in a cheaper product, which also provides the flexibility of incorporating only the required sensors. The cheapest glove from 5DT company costs USD 995 [3], whereas the construction of the glove designed in this work required less than USD 160. Although quite a few research works have been performed regarding SLR using data gloves [11, 15, 20, 22], very few of them attempted to provide a complete solution – starting with the construction of data gloves to continuous detection of sign language, ensuring complete character set recognition of a particular alphabet.

Patil et al. [20] attempted to classify all 26 characters of the ASL alphabet by detecting bending only. Section 3.1 explains why this is not a feasible solution. They tried to classify letters based on the ranges of flex sensor values without any specific direction on how to decide the range values. Elmahgiubi et al. [11] elaborately described a procedure of constructing a data glove capable of detecting 20 letters of the ASL alphabet. However, just like the previous work, they did not clearly specify how to find with the boundary values to perform ranged queries. None of these works provide any method of sign language modeling, i.e. retrieving intelligible messages from a stream of data. Mehdi et al. [15] used 7-sensor glove of 5DT company to collect data and subsequently used Artificial Neural Network to classify signs. They left out two dynamic letters of the ASL alphabet and tried to detect the rest 24 letters. Although they discuss briefly sampling rate of data collection from sensors, they also do not elaborate on sign language modeling. SLARTI is a system developed at the University of Tasmania to recognize Australian Sign Language (Auslan) [22]. This system exploits the fact that signs can be described in terms of four basic manual features: (i) handshape, (ii) orientation, (iii) place of articulation, and (iv) motion. SLARTI contains four feature-extraction neural networks, one for each feature.

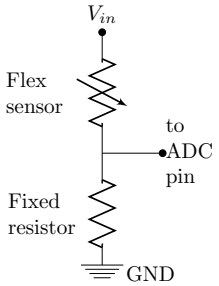


Figure 3: Voltage divider circuit.

3 METHODOLOGY

In this section, we discuss several design and implementation issues of the sign language recognition system. A sign language recognition system can be subdivided into a series of discrete steps. The first step is to identify the necessary sensors and their suitable positions in the gloves to correctly capture different hand gestures. The glove is then constructed by embedding sensors at the desired positions. With the constructed glove a significant amount of data are needed to be collected which are fed to train a Machine Learning (ML) model to classify the captured data. Detailed study of the dataset might reveal whether a further modification to the glove and relative positions of the sensors is necessary or not. If a modification is required, all or a few of the previous steps need to be repeated. Once a suitable model is trained, *sign language modeling*, that is, determining intelligible messages from a stream of sensor data starts. The trained model can then be deployed in the production environment. Further experiments need to be conducted to find out the accuracy of the model and to gain insight on comparative studies with other such models. Different issues of each of the steps are described next.

3.1 Data Glove Construction

A number of options are available for constructing the data glove. A normal woolen glove can be used as the base. However, as woolen gloves tend to be roomy and do not fit tightly around the hands, other types of gloves are preferable. Lycra gloves, wicketkeeper's inner gloves, etc. are all good candidates. Surgical gloves can be used too, but as these tend to fit too tightly around the hand, it might become cumbersome to wear and take the glove off repeatedly with all the sensors attached to it. As the same glove would be used by people with varying hand sizes, an average sized glove is preferred.

Detection of Bending. The primary criteria to classify symbols in ASL and BdSL (or any other sign language) is the *bending* of the individual fingers. To detect this bending a type of resistive sensor called *flex* sensor can be used. Flex sensors are made of a material which changes its resistance upon bending. In particular, the resistance increases as the flex sensors are bent. Separate five flex sensors are required to detect the bending of five different fingers. The flex sensors are attached to the back of each finger. To facilitate ease of removal and attachment of sensors, pockets using strips of cloth can be sewn at the back first. As microcontrollers cannot measure resistance directly, a voltage divider circuit is required. Interestingly, the exact resistance value of a flex sensor at a certain instance is not required, rather a measure of change in resistance serves the purpose. The more the sensor is bent, the more its resistance increases. Therefore, by simply observing the relative changes in resistance value, it can be inferred whether the sensor is straight or bent. The voltage divider circuit shown in Fig. 3 has been used. When the flex sensor is bent, its resistance increases, which results in lower voltage in Analog-to-Digital Converter (ADC) pin. If the flex sensor is subsequently straightened, its resistance decreases, which results in higher voltage in ADC pin. The circuit in Fig. 3 can be arranged in an alternative form where the position of the flex sensor and the fixed resistor is interchanged. In this case, the ADC pin will receive a relatively high voltage when flex sensor is straight and a relatively low voltage when the sensor is bent.

Detection of Contact. Although just by determining the relative bending of each finger a number of symbols can be classified, there exist a few symbols that cannot be classified solely based on this information. For example, the ASL characters 'M', 'N', and 'T' shown in Fig. 1 have similar bending of the fingers. In case of 'M', the thumb goes between the little finger and the ring finger, in case of 'N' the thumb goes between the ring finger and the middle finger, and in case of 'T' the thumb goes between the middle finger and the index finger. In order to differentiate these letters the contact information between the thumb and other fingers is needed. Similarly, the letters 'R', 'U', and 'V' also have similar bending of individual fingers but varies in contact points.

The contact information between two points can be captured in several ways. One option is to use a force sensing resistor (FSR). When pressure is exerted upon the conductive film of an FSR, its resistance decreases. Thus it is a resistive sensor and a voltage divider circuit similar to the one shown in Fig. 3 can be used. However, there are certain caveats which are not apparent at the first look. First of all, the area occupied by the conductive film is not that large. As a result, it takes a bit of time for the user to get habituated with pressing the exact area. Although this situation can be alleviated to some extent by using a larger FSR, a similar problem still remains for users with below average hand size. As the glove tends to feel more roomy for such users, the conductive strip tends to slip from the initial position. A more severe drawback can be observed while trying to distinguish 'M', 'N', and 'T'. The most suitable zones to place FSR to detect these letters would be the side of the little finger, side of the ring finger, and side of the middle finger, as evident from Fig. 1. Because during signing 'M' the thumb will be directly on the FSR on the little finger, the resistance of this particular FSR will be lower than those of the rest two. In reality, the lateral pressure exerted by the middle finger on the FSR of the ring finger, and that

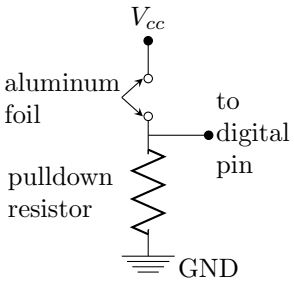


Figure 4: Simulating button using aluminum foil.

by the index finger on the FSR of the middle finger is very close to the pressure exerted by the thumb on the FSR of the little finger. Therefore it becomes difficult to distinguish these letters. Similar arguments for 'N' and 'T' rules out the possibility of using FSR as a contact sensor.

The most viable option to capture contact information is to imagine the contact sensors simply as *buttons*. If there is a contact i.e. the button is ON, then a particular signal level is generated. If there is no contact i.e. the button is OFF, then another level of signal is generated. Instead of an actual button, the two ends of the button can easily be simulated by conductive surfaces. A cheap and readily available solution could be obtained from aluminum foils used as chocolate wrappers as shown in Fig. 4. One drawback of using such foils is that over time they tend to form creases at certain points due to bending and lose conduction at the points of crease formation. Other than aluminum foils, conductive fabrics or conductive threads can also be used as contact sensors. The final implementation of the glove in this work has used conductive fabric.

Detection of Orientation and Motion. Apart from bending of fingers and contact between two fingers, another feature that is sometimes required is the orientation of the hand. For example, the ASL letters 'H' and 'U' as observed from Fig. 1 have similar bending and contact, the only difference is their orientation. Similar observations can also be made for the letter pairs $\langle G, Q \rangle$ and $\langle K, P \rangle$.

One last feature required to differentiate certain characters is motion. Although most characters in ASL and BdSL are static, very few are dynamic, meaning motion is involved when they are signed. For example, the ASL characters 'J' and 'Z' require certain types of motions as shown in Fig. 1. To detect orientation and motion, an accelerometer and a gyroscope unit need to be used. A suitable candidate is the MPU6050 board, which is capable of detecting six degrees of freedom, namely gravitational force along the three axes and the rotational speed along the three axes. The module has the direction of the two axes drawn on it, the third one can be inferred from the two. It can communicate with microcontrollers via the I2C protocol. The best place to attach the module would be back of the hand, a few centimeters above the wrist.

3.2 Data Collection

Data has been collected from five users with varying hand sizes using the same glove. Table 1 shows the hand sizes and comments for the five participants. To measure the hand size, two metrics

Table 1: Hand size of participants.

User ID	Length	Circumference	User comment on glove
1	7.0	6.5	Slightly roomy
2	6.5	6.1	Too large
3	7.2	7.2	Too tight
4	6.9	6.8	Comfortable
5	7.0	7.0	Perfect fit

have been used – one is length, another is circumference. Both of these parameters have been recorded for each user.

To facilitate the process of data collection and later on real-time evaluation, a desktop application was developed. Data can be collected in two ways: 1) data can be logged to an SD card connected to the microcontroller board and subsequently moved to a computer, or 2) the microcontroller can be connected directly to the computer via USB port where the data can be stored directly. The second method has the obvious advantage that a manual trip from a microcontroller to a computer can be saved. Each user gave data in two sessions for each language (ASL and BdSL). In each session, the user gave 500 data for each character. Therefore, for each character, there are 1000 data points from each user. There are some common sings between the ASL and BdSL alphabet, they have been collected once. In each session, a user gave data for one character in five *iteration*. An iteration consists of the user holding the sign for ten seconds. Data was collected at 100ms interval, resulting in 100 data in one iteration. Between successive iterations, the user straightened his/her hand once and then made a fist with his/her hand once before moving onto the next iteration. This has been done so that the previous iteration does not have any effect on the next iteration. This ensures variability in data which would be observed when a user is signing different letters continuously. In case of dynamic signs, the stationary state of the sign has been recorded. As 'I' and 'J' have the same static configuration of hand, the dataset contains the data of 'I' but not that of 'J', 'J' being the dynamic one.

Each feature vector consists of sixteen features. The first five features, namely *thumbFlex*, *indexFlex*, *middleFlex*, *ringFlex*, and *littleFlex*, correspond to the values of five flex sensors attached to five fingers. These are required to detect bending of each of the fingers. The next five features are for five contact sensors placed at various locations of the hand. The locations were chosen by studying the symbols through an iterative approach to data collection and classification. A contact sensor was placed at a suitable location only if more than one character showed similar bending but varied in contact. Five places were identified to be enough to distinguish among the characters of ASL/BdSL. The five features corresponding to these places are *indexFront*, *middleFront*, *middleSide*, *ringSide*, and *littleSide*. The next three features correspond to the X, Y, and Z component of the accelerometer. It was observed that certain pairs of characters have similar reading for flex sensors and contact sensors. However, they differ in orientation. Accelerometer was used to detect this difference in orientation. The three features next to that correspond to the X, Y, and Z component of the gyroscope. There are a few dynamic characters in ASL and BdSL. The motions involved in these dynamic characters can be detected using gyroscope. The last three columns in the dataset correspond to the letter being signed i.e. the label, the user ID, and the session ID for that particular user respectively. As the difference of data over users and sessions has been studied in details in Section 5.1, user ID and session ID have been included in the dataset. This amounts to a total of nineteen columns in the dataset, among which sixteen are features. The full dataset can be accessed at [6].

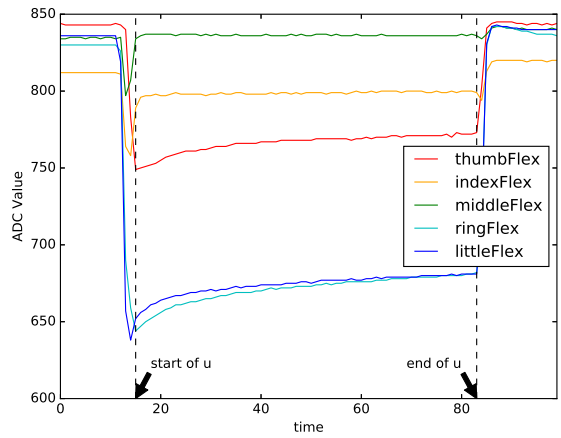
The necessity of various sensors described in Section 3.1, and in turn the selected features, can be more succinctly understood with the help of certain features observed together. For example, 'U' and 'V' have similar bending for all the fingers and thus are not separable based solely on flex sensor values. However, the index finger and the middle finger come in contact while signing 'U', but not while signing 'V'. That is, the contact sensor positioned between these two fingers give a value of 1 for 'U' but 0 for 'V', while the flex sensor values remain similar for both the characters. This phenomenon is demonstrated in Figure 5. The plots were generated by keeping all of the fingers straight and apart from each other for some time, then moving on to the symbol of a character and holding it for some time, followed by a fully straight hand again. Similarly, the use of accelerometer can be justified with cases such as 'H' and 'U' as shown in Figure 6. In case of these two letters, the bending of the fingers are similar but the orientation is different. This difference in orientation is apparent from the X component of the accelerometer reading. Finally, the role of gyroscope in detecting motion for dynamic characters such as 'Z' is evident from Figure 7. The beginning and the ending of the plot corresponds to the static 'Z' symbol, while the middle portion corresponds to the z-shaped motion. As can be seen from the figure, the flex sensor values remain similar throughout the time period. However, a significant amount of fluctuation is observed in gyroscope reading during the motion. The points corresponding to the time when first horizontal motion, the diagonal motion, and the second horizontal motion was detected have been labeled.

3.3 Training a Classifier

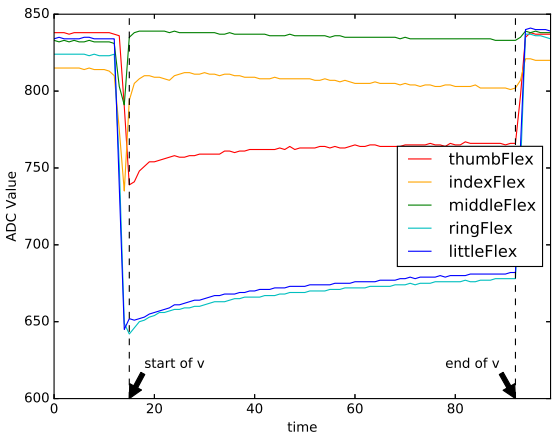
Once data has been collected, the next step is to train a suitable classifier. As the model should be able to work in a resource-constrained environment, not all classifiers are suitable. For example, lazy learners like KNN are out of consideration. On the other hand, neural networks have an added advantage that the model can be trained on a different processor and only the weights can be simply loaded into the microcontroller. Due to this fact, if a neural network based classifier requires heavy computation during training phase it is not a concern. But if the testing phase requires heavy computation, that classifier should be avoided.

Although at first glance one might assume that there should be a single classifier for the two languages (ASL and BdSL), in reality, two different classifiers are required. This is because there are eleven overlapping symbols between the two languages. There is an option in the system for the user to choose the desired language.

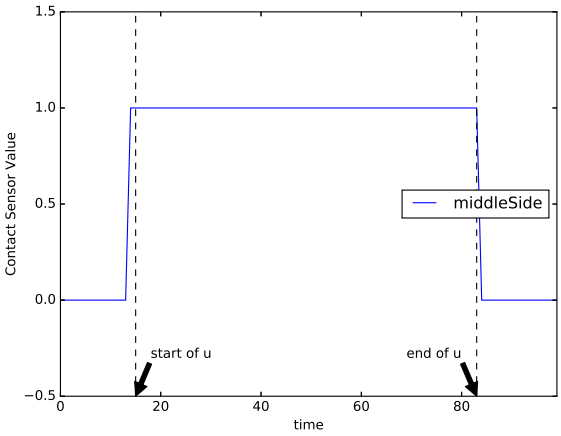
For classification purpose, data can be partitioned in a number of ways. For example, the data of a single user can be treated as the training data, the rest of the data can be treated as test data. An opposite extreme can be treating a certain percentage of the



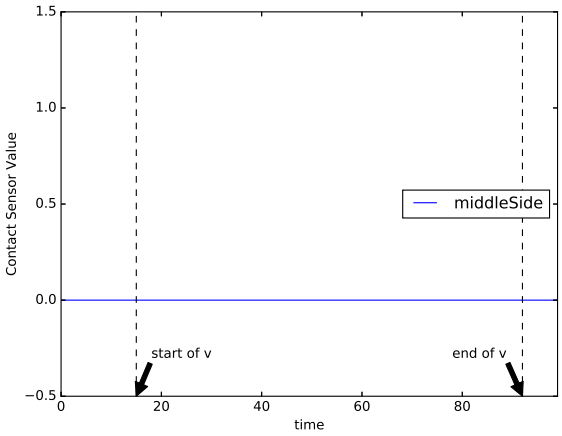
(a) Linechart of flex sensors for 'U'.



(b) Linechart of flex sensors for 'V'.



(c) Linechart of middleSide contact for 'U'.



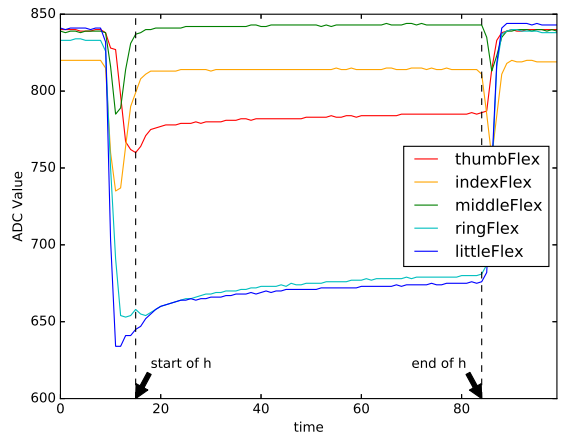
(d) Linechart of middleSide contact for 'V'.

Figure 5: Linecharts of flex sensors and the contact sensor between the index finger and the middle finger for 'U' and 'V'.

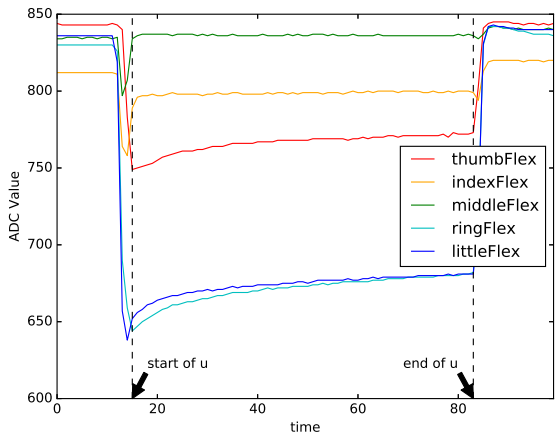
data of each user as training, and the rest as test. The latter option would give better performance over a range of users, as the trained model would have better generalization capability. The former model might have better performance for that individual but would not provide good results over a vast range of users. Moreover, as the glove should have the capability to work for users who have not experimented with the glove, a better option would be to use the data of a group of users as training while treating the data of others as testing.

While training, the classifier is trained based on the values from the five flex sensors only. In other words, an initial classification is made based on the bending of the fingers. This essentially puts a feature vector in a cluster, and if the cluster contains more than one letter, a rule-based classification is employed. The exact rule can be found out by simply fitting a decision tree on the values of contact sensor and accelerometer. For example, the letters 'V' and 'U' have a similar type of bending of the fingers. So 'V' can be classified as 'U' and vice versa by the classifier, i.e., the classifier puts it in a cluster containing two letters: 'V' and 'U'. But a contact sensor between the index and middle fingers allow the rule-based classifier to distinguish between these two characters. A block diagram of the classification subsystem is presented in Fig. 8.

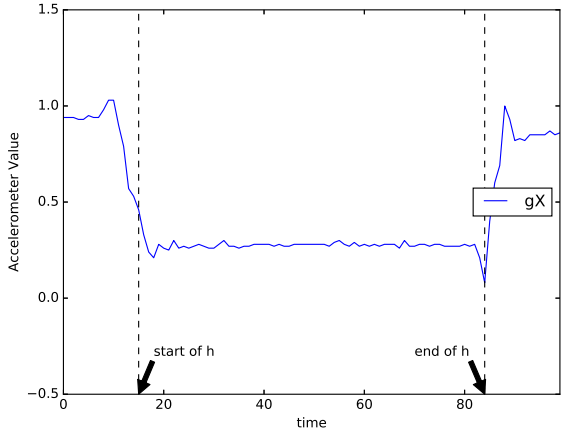
It is worth noting that instead of multi-level classification, an end-to-end classification approach can be employed too, i.e., feeding all of the data from flex sensors, contact sensors, and accelerometer. However, the former approach was employed in initial implementation and later kept unchanged for a few reasons. Firstly, the glove construction and the training steps are closely coupled in this work. When it was observed that flex sensors alone were not sufficient for classification, only then the other sensors were incorporated in the glove. As the contact sensors provide a value of 0 or 1 and the accelerometer mostly provides a value within [-1,1], intuitively it made more sense to pass the sensor readings through a decision tree to break any sort of tie. Secondly, this approach, in general, uses



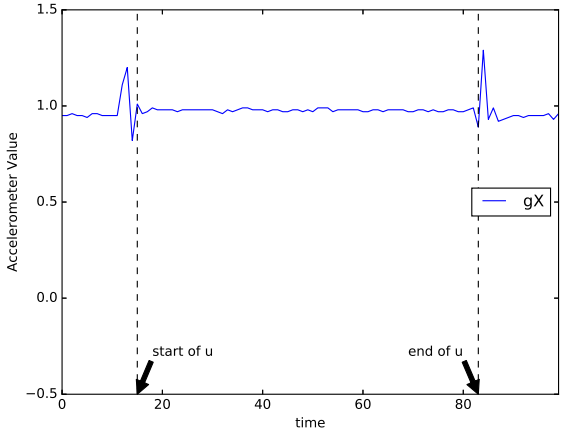
(a) Linechart of flex sensors for 'H'.



(b) Linechart of flex sensors for 'U'.



(c) Linechart of gX for 'H'.



(d) Linechart of gX for 'U'.

Figure 6: Linecharts of flex sensors and X component of accelerometer for 'H' and 'U'.

a minimal amount of data during computation. At first, only the flex sensor values are used for computation during classification, data from other sensors are used only when required. Finally, it was observed that an end-to-end classifier did not affect the accuracy considerably. There was a very slight deterioration in accuracy percentage for an end-to-end classifier, the difference being in the hundredths place.

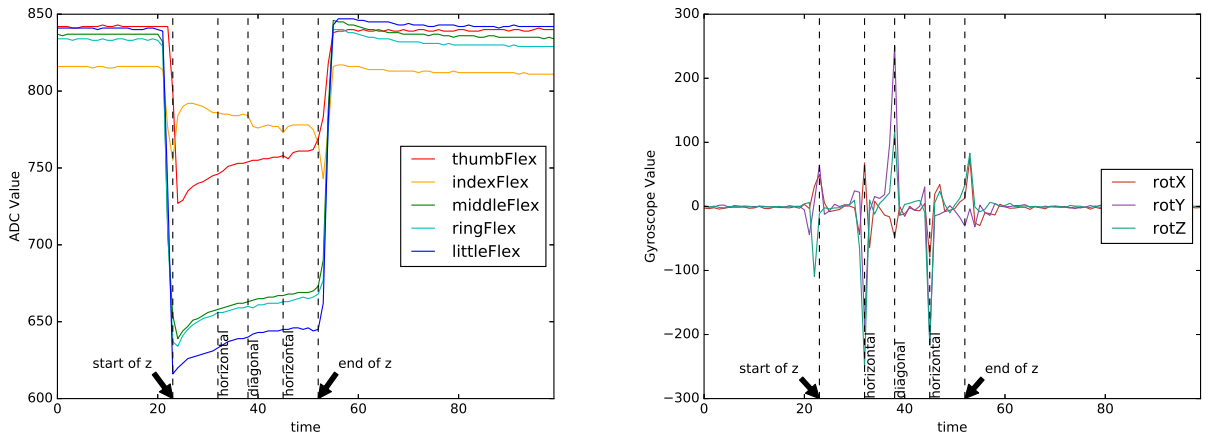
Once a classifier has been trained on the collected data, it can be deployed in production environment. Experiments can then be performed to analyze the performance of the system and to conduct comparative studies.

4 SYSTEM DESIGN AND PERFORMANCE TUNING

This section discusses design issues of the developed system. Section 4.1 discusses interfacing among the various components of the system. Section 4.3 presents a simple technique for *sign language modeling*, i.e. the process of forming intelligible messages from a stream of sensor values. The developed technique is capable of filtering out the transitional noise observed when the user moves from letter to letter. A suitable Machine Learning algorithm has been chosen from four after a comparative study for the final deployment of the system. The four algorithms are Artificial Neural Network (ANN), Support Vector Machine (SVM), Ensemble Learning (Random Forest), and K-Nearest Neighbor (KNN). Section 4.5 discusses suitable parameter selection for these algorithms.

4.1 Physical Connectivity

The constructed glove is interfaced to an Arduino Mega microcontroller board via circuitry concised in a veroboard. The sensor data collected by the microcontroller is then sent over to a computer via the USB port. In case of Arduino Mega, there is already a USB to serial converter which facilitates this communication. If however, a standalone microcontroller is used lacking this capability, a



(a) Linechart of flex sensors for 'Z'.

(b) Linechart of gyroscope values for 'Z'.

Figure 7: Linechart of values from flex sensors and gyroscope for 'Z'.

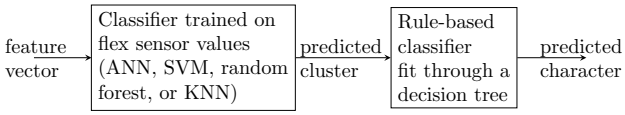


Figure 8: Block diagram of the classification subsystem.

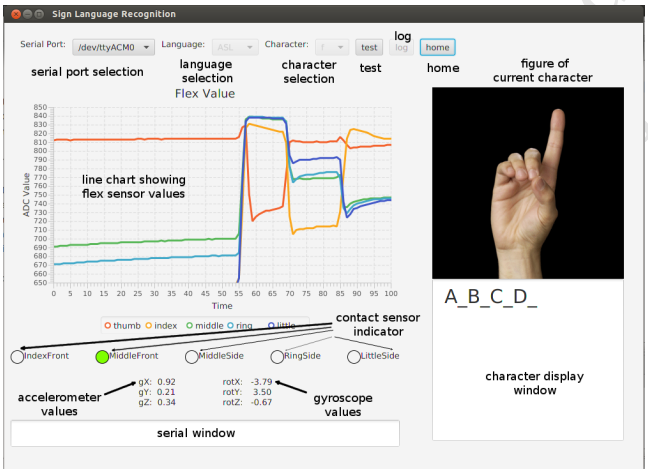


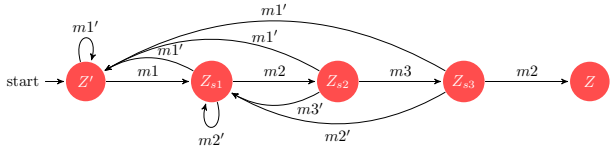
Figure 9: Labeled real time assessment window.

separate chip has to be used for this communication. The required visualization can then be presented on the computer monitor. A variant of this setup might be to altogether get rid of the computer and perform all the functionalities in the microcontroller. That is also possible but would require some extra components. For data collection purpose the microcontroller would require a Micro SD card/SD card breakout board along with the card, and for display purpose, it would require a display module. Two common choices for display modules are LCD and OLED, the former option being the cheaper one. Alternatively, the microcontroller could communicate to a smartphone via Bluetooth.

A desktop application was developed in Java to facilitate data collection and real-time assessment. A labeled diagram of the real-time assessment window is presented in Fig. 9. The line chart shows the ADC values as affected by the bending/straightening of flex sensors. There are five lines for each of the flex sensors attached to the five fingers. Just below the line chart are five indicators representing whether the contact sensors are in contact or not. Next, the accelerometer and gyroscope readings are dumped in a tabular fashion. At the very bottom, there is a serial display which shows a set of sensor values once the test button situated at the top row is pressed. The window contains a couple of dropdowns to select the language and character. The latter dropdown is used only in data collection mode, but not in real time assessment mode. The log button is also exclusively used in data collection mode. On the top-right, there is an image which contains the letter being signed at that instant. Just below that there is a window showing the stream of characters as classified by the trained classifier. The underscores represent space in this display. An explicit character has been used to represent space so that it is obvious exactly when the system introduces a pause after the last character being recognized. If a user continues to give sign without sufficient pause these will not be displayed, following the convention of fingerspelling. This window clears itself once it gets full.

4.2 Detection of Dynamic Symbols

Both ASL and BdSL alphabet have a few dynamic letters. A simple way to detect dynamic symbols is to map them to finite state machines (FSM). As 'Z' is the most complex among all of the dynamic symbols, the detection of 'Z' is explained in details. Other dynamic symbols can be similarly mapped to corresponding state machines. To express 'Z', a user needs to first make the static 'Z' symbol as shown in Fig. 1. While holding this symbol, the user



$m1$: the 'z' symbol
$m2$: rightward movement while holding $m1$ within an interval t of previous state
$m3$: diagonally downward movement while holding $m1$ within an interval t of previous state
$m1'$: any symbol other than 'z'
$m2'$: any movement other than $m2$ while holding 'z'
$m3'$: any movement other than $m3$ while holding 'z'

Figure 10: FSM to detect 'Z'.

needs to make one rightward movement, followed by a diagonally downward movement, then finally another rightward movement. One movement must follow the previous one within a certain time limit t, i.e. everything is done in one swift motion. Fig. 10 shows a finite state machine to detect 'Z'. Here the state Z' represents any symbol other than 'Z', state Z_{s1} represents the static 'Z' symbol, state Z_{s2} represents 'Z' upto first rightward movement, state Z_{s3} represents 'Z' upto diagonally downward movement, and finally Z represents successful detection of 'Z'. The rightward and diagonally downward movements can be detected by studying the current and a few prior sampled values of the gyroscope.

4.3 Sign Language Modeling

Sign language modeling using data glove is a field which has not received that much attention from the research community. Data is collected from the sensors at an interval of 100ms. For users with fast response, this interval can be changed. At this rate, a classifier makes a classification as soon as the sensor data is fed to it. But right after the classifier makes a prediction that the current symbol represents a character X (here X is a placeholder), it is not displayed by the system. Rather the system would wait till it sees a number equal to a *MINIMUM_THRESHOLD* of consecutive Xs. This has been done to remove the effect of noise during transitions from one letter to another. The value of this *MINIMUM_THRESHOLD* might again depend on the speed of an individual. While using fingerspelling, a space or division between words is represented by holding a symbol longer than usual. Therefore, another parameter, *MAXIMUM_THRESHOLD*, has been introduced. If a symbol is continuously observed for more than or equal to this number, a space is introduced by the system. An obvious question is how to show double letters. In fingerspelling, a double letter is shown by giving a slight bump of the hand, or a slight motion to the side. This bump/motion can be detected with the help of gyroscope. So if a bump is observed before *MAXIMUM_THRESHOLD*, the system could display a double letter. However, it has been observed that the readings shown by gyroscope while forming a bump correspond to similar readings shown during certain transitions from character to character. This resulted in the system to show double characters when in fact there was only a transition. Hence this feature was left out. The modeling algorithm keeps track of a variable, *COUNTER*, which represents the number of contiguous steps the current character has been predicted. When the current prediction X is observed to be different than the previous prediction X', the *COUNTER* is reset to one. In the current system to represent double letter the user needs to deliberately create some transitional noise in order to reset the *COUNTER*.

Fig. 11 shows a simulation of this sign language modeling in action. Here the *MINIMUM_THRESHOLD* is set to two and the *MAXIMUM_THRESHOLD* is set to four. When the first 'A' is observed, the *COUNTER* is set to one. The second 'A' increases the *COUNTER* to two. As the *COUNTER* is now equal to the *MINIMUM_THRESHOLD*, an 'A' is displayed (step 2). Next, another 'A' comes up, increasing the *COUNTER* to three. Then another 'A' increases the *COUNTER* further to four. As the *COUNTER* is equal to the *MAXIMUM_THRESHOLD*, a space is inserted in display (step 4). Another 'A' after that increases the *COUNTER* to five. Now a 'C' comes up. As the current prediction 'C' is not equal to the previous prediction 'A', the *COUNTER* is reset to one. Next, a 'B' comes up, resetting the *COUNTER* to one again. Another 'B' increases the *COUNTER* to two, making it equal to *MINIMUM_THRESHOLD*, so a 'B' is displayed (step 8). It is worth noting that the transitional noise 'C' was filtered out by using this simple mechanism.

4.4 System Integration

Figure 12 provides an overview of the full system. Data is read from the glove at a definite sampling rate by the processing unit. Apart from the software support to communicate with the glove and the display unit, the processing unit contains a previously trained model of a classifier and an implementation of the sign language modeling algorithm described in Section 4.3. The classifier is trained using previously collected labeled data as described in Section 3.3. However, the classifier model and the algorithm comes into play only during real-time analysis, these are not necessary for the data collection mode. If the system is operating in data collection mode, the processing unit sends the data to a computer (which can also act as a display unit) for storage and the successive training of a classifier. If the system is operating in real-time classification mode, the processing unit performs two extra steps before sending data to the display unit. Firstly, it is going to classify the set of sensor values just read in into one of the characters using the pre-trained model of a suitable classifier. Secondly, it is going to input the classified character to the algorithm of sign language modeling. The sign language modeling algorithm outputs the character to be displayed depending on the recent few classified characters as described in Section 4.3. Thus a stream of sensor values sampled at a definite rate first gets classified by a pre-trained classifier to generate a stream of characters, then the sign language modeling algorithm dictates which character should be displayed depending on that stream of characters. Finally, the display unit shows the actually recognized character.

4.5 Sensitivity Analysis

Each of the studied algorithms has one or more tunable parameters. To fix a parameter of an algorithm, the data was randomly split into 80% train data and 20% test data. It was ensured during the split that the proportion of data from user X for character Y were equal for all X and Y. Then the same algorithm was run over the split by varying the value of the parameter. The value corresponding to the highest accuracy was fixed for that parameter.

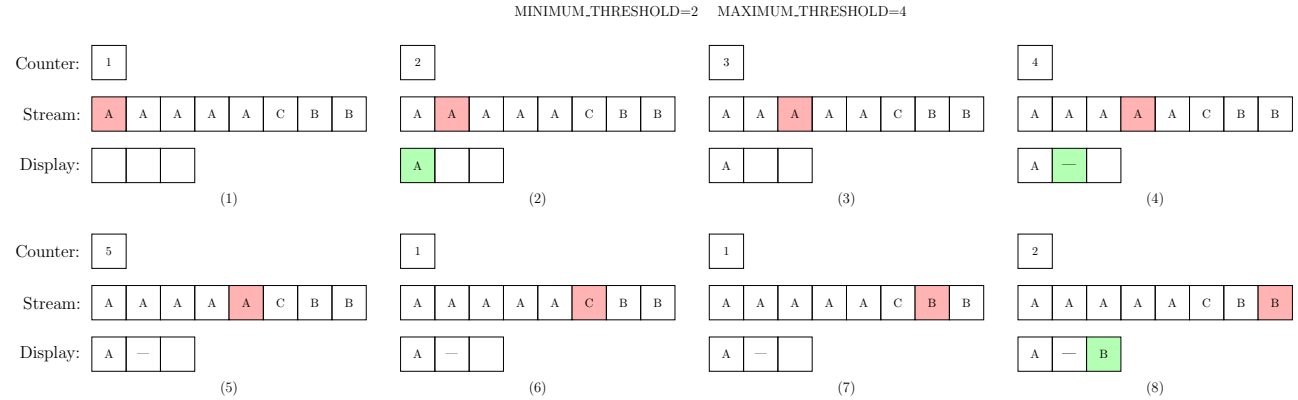


Figure 11: A simulation of sign language modeling.

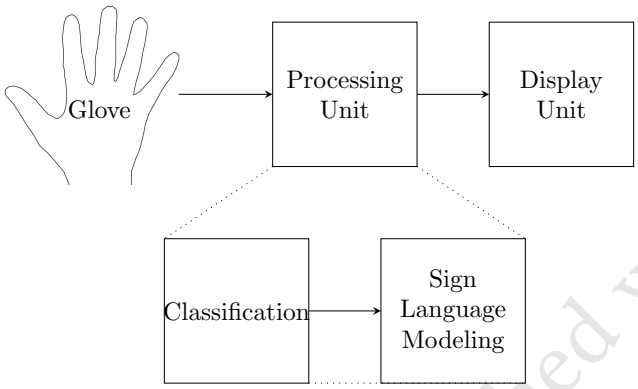


Figure 12: Block diagram of the full system.

In case of ANN, a single hidden layer was used. Studying the effect of increasing the number of layers was ruled out considering the involved time and space complexity. It was observed that accuracy increased steadily as the number of nodes in hidden layer was increased up to 7 (24) for ASL (BdSL). After that, the accuracy oscillates around a certain mean. Four activation functions were studied – tanh, logistic, identity, and relu. It was observed that logistic function showed the best accuracy for both ASL and BdSL. In case of SVM, Radial Basis Function (RBF) showed better accuracy compared to linear, poly, and sigmoid. In case of random forest, the accuracy increased steadily as the number of stumps was increased up to 40 (32) for ASL (BdSL). After that, the accuracy oscillated erratically. In case of KNN, the accuracy did not considerably change after increasing the value of K beyond one for ASL. On the other hand, the accuracy varied wildly for BdSL, exhibiting a peak for K=200. A summary of the different parameters for the four algorithms is presented in Table 2.

5 EXPERIMENTAL RESULTS

Experiments were conducted on the dataset to understand the composition of data and to analyze the performance of various algorithms on the data. As the data was collected from multiple users in different sessions, one question of interest was whether

Table 2: Chosen values of different parameters.

Algorithm	Parameter	ASL	BdSL
ANN	no. of nodes in hidden layer	7	24
ANN	activation function	logistic	logistic
SVM	kernel function	RBF	RBF
Random Forest	number of stumps	40	32
KNN	no. of neighbors K	3	200

the data between any two users (or two sessions of a user) are statistically significantly different. This question is addressed in Section 5.1. At first take it might seem like a simple range-based query over the flex sensor data might be sufficient to classify the letters. Section 5.2 elaborates why range-based query does not give good results. Section 5.3 discusses the relative nonuniformity in bending of fingers over different users. Finally, Section 5.4 compares the performance of four algorithms over the dataset.

5.1 Difference of Data over Users and Sessions

To answer the question whether the data between any two users (or two sessions of a user) are statistically significantly different, Wilcoxon Signed Rank Test was conducted at 0.05 significance level. It was observed that all pairs of users showed statistically significant difference in at least one of the five flex sensors. Among the five users, three showed statistically significant difference in at least one of the five flex sensors between different sessions. Among the two users whose data did not exhibit statistically significant difference over sessions, one user opined that the glove fit perfectly whereas the other user stated the glove was comfortable. On the other hand, the greatest difference over sessions was observed for the user with the smallest hand size. This conforms to the intuitive observation that every time a user with a small hand size wears the glove, the sensors tend to sit on his/her hand a bit differently. Therefore, the readings would be different each time. Conversely, a user with a good fit will experience similar positioning of the sensors over different sessions. As a result his/her data would not exhibit significant difference over different sessions.

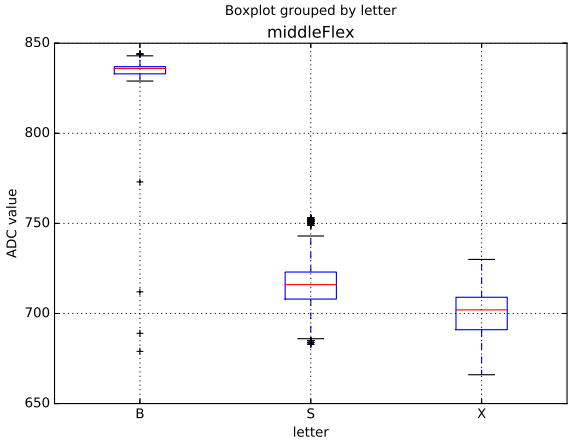


Figure 13: Boxplot of middle flex values for 'B', 'X', and 'S'.

5.2 Infeasibility of Range-Based Query

Classification was first attempted through a range based query. However, it was observed that not all characters can be differentiated by simply observing the range of flex sensor values. Fig. 13 shows boxplot of flex sensor values on the middle finger for letters 'B', 'S', and 'X'. A brief look at the plot reveals that there is a clear distinction between the range of values for 'B' and 'S'. The same can be said for the pair 'B' and 'X'. However, 'S' and 'X' have a considerable amount of overlap over their full range. Similar trends can be observed for the other flex sensor values. Moreover, 'S' and 'X' are such a pair which cannot be differentiated with the help of existing contact sensors, they must be differentiated with the help of the flex sensor values. Therefore a more elegant solution is required for classification. Later classifications were attempted using various Machine Learning algorithms which exhibited superior performance.

5.3 Nonuniformity in Relative Bending of Fingers

One interesting observation of this study is that the same trend in relative bending of the fingers of different users can not be observed for all letters, thus rendering an initial calibration ineffective. Fig. 14a and Fig. 14b shows the linecharts of mean values of flex sensors for all five users for the letters 'A' and 'P' respectively. From the linechart of 'A', it is evident that the little finger of User 4 experiences less bending compared to that of other users. Intuitively, it might seem like this should be the case for all other letters. However, the linechart of 'P' reveals one of the many counter examples. Here it is observed that the little finger of User 4 actually experiences the second most bending among the five users. Therefore, no general comment can be made regarding the relative bending of different users.

5.4 Performance of Different Algorithms

A K-Fold Cross-Validation was performed on the dataset by taking ten folds. Essentially the dataset was divided into ten sets, and training and testing was performed ten times. In each of the iteration one set was taken as the test set, and the rest nine sets were taken collectively as the train set. The advantage of K-Fold Cross-Validation over multiple random splits is that every data point is guaranteed to be in the train set at least once, the same guarantee being provided for the test set too. The average accuracy over the ten iterations is presented in Table 3. It can be observed from the presented accuracies that average accuracy of random forest and KNN are clearly better than SVM. The performance of ANN is slightly below that of random forest and KNN. However, as KNN is a lazy learner, it is not suitable for embedded environments. Random forest could be deployed in an embedded environment, but between a trained model of ANN and a trained model of random forest, ANN has been found to be faster. Therefore ANN might be taken as the overall best choice. Fig. 15 and Fig. 16 shows the color coded confusion matrix resulted from running ANN on a random 80-20 split of ASL and BdSL dataset respectively.

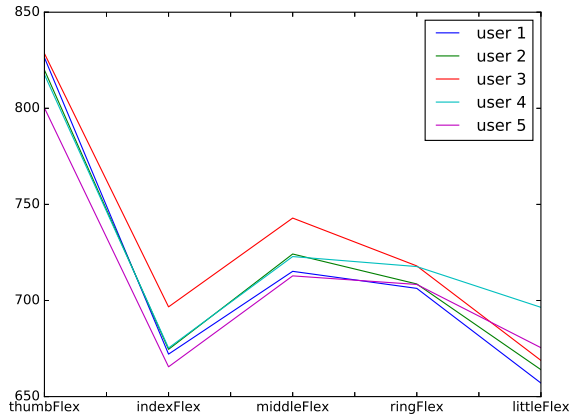
Table 3: Average accuracy of algorithms after 10-Fold Cross Validation.

Algorithm	Accuracy (%)	
	ASL	**BdSL**
KNN	96.1448	93.8643
Random Forest	96.1352	96.3422
ANN	95.8784	93.4368
SVM	94.916	92.1859

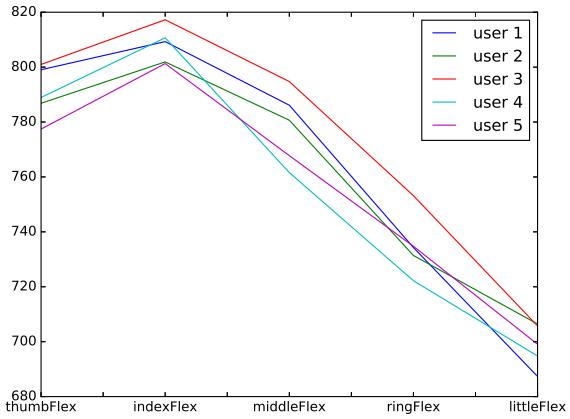
A trained ANN model was deployed in the final system. It was observed that the final accuracy of the system for fingerspelling recognition was around 96%, closely following the performance of the classifier over the collected data. However, the two threshold values required by the sign language modeling algorithm had to be manually tweaked for different users; as otherwise spaces were being introduced for comparatively slower users before they expected, and some letters were missed by the system for comparatively faster users. As a future work, a calibration phase can be added to automatically set the threshold values for different users.

5.5 Classification of Non-Standard Gestures

Currently, it is possible to confuse the system with certain non-standard gestures which might look like one character to the naked eye but is actually classified as another character by the system. The main reason behind this can be thought of as the non-availability of sensor values corresponding to these gestures during the training phase of the classifier. However, it is not feasible to collect data for all possible non-standard gestures. This problem can be solved to some extent by generating some synthetic data. Generation of synthetic data and its subsequent use in training a classifier has been extensively studied in the literature [12, 23, 24]. In future, this strategy can also be employed for this work. It has also been observed that in many cases the system outputs the correct result even when the gestures are deformed to some extent, and starts to show different result only when the gesture becomes ineligible to the naked eye too. As an example of a confusing case of non-standard gesture, flex sensor values corresponding to some gestures of the letter 'I' have been presented in Figure 17. In case of 'I', all the



(a) Linechart for 'A'.



(b) Linechart for 'P'.

Figure 14: Linecharts of mean values of flex sensors of different users for 'A' and 'P'.

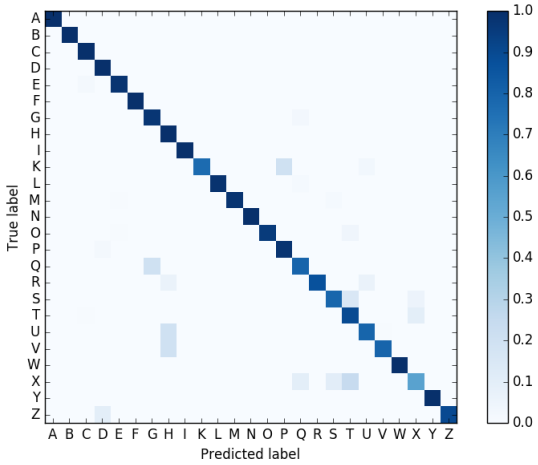


Figure 15: Confusion matrix for ANN run on a random 80-20 split of ASL dataset.

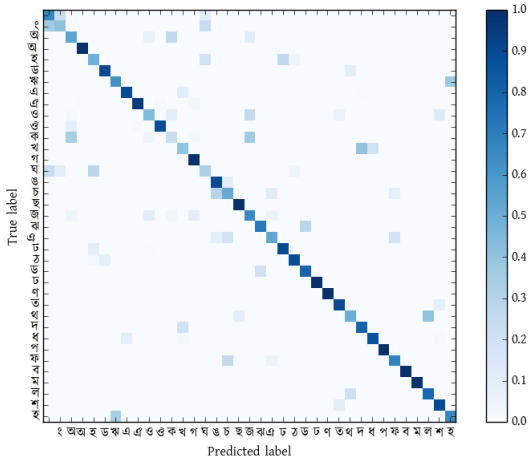


Figure 16: Confusion matrix for ANN run on a random 80-20 split of BdSL dataset.

fingers except the little finger are bent, the thumb is not bent fully but bent enough to cross in front of the index finger. The first half or so of the plot was obtained from this standard gesture. After that, the thumb was gradually moved to the side so that it lost contact with the index finger but still looked close to it. In this condition, the gesture might still look like 'I' to the naked eye. However, the system recognized this gesture as 'Y'. In standard gesture of 'Y', the thumb is separated from the index as far as possible. The exact position from where the system started to detect 'Y' has been marked with a dashed vertical line in the plot.

5.6 Comparison with Previous Works

A comprehensive comparison among the previous works on sign language recognition is presented in Table 4. Most of the previous works were on signed letter recognition, whereas Naoum-Sawaya et al. [17] and Vamplew et al. [22] attempted signed word recognition i.e. words that can be expressed using a single gesture. The former used DIP based method whereas the latter used glove based method. The more complex case of sign language modeling using DIP based method was covered by Liwicki et al. [14] and Rekha et al. [21]. However, to the best of our knowledge, sign language modeling using data gloves has not been studied so far, which this work does. Moreover, the variability in data collected from different users has not been studied extensively by the previous works. On the other hand, this work makes a detailed observation in this aspect. Although some works reported recognition of dynamic characters, they did not clarify the underlying mechanism. This work provides a detailed explanation of an FSM based method of dynamic character recognition.

Table 4: Comparison with previous works.

Author	Language	Symbol		Method	Algorithm	Accuracy	
		Letters left out	No. of finger-spelt words			Recognition	Modeling
Akmeliawati et al. [9]	Malayasian	-	-	DIP	ANN	95.67%	-
Kuznetsova et al. [13]	ASL	J,Z	-	DIP	multi-layered random forest	97.55%	-
Dogic et al. [18]	Bosnian	-	-	DIP	ANN	84.00%	-
Dong et al. [10]	ASL	J,Z	-	DIP	random forest	90.00%	-
Nabiyev et al. [16]	Turkish	21 consonants	-	DIP	ANN	96.00%	-
Pandey et al. [19]	ASL	-	-	DIP	ANN	90.00%	-
Liwicki et al. [14]	BSL	-	100	DIP	multi-class logistic regression / HMM	84.10%	98.90%
Naoum-Sawaya et al. [17]	ASL	N/A	N/A	DIP	CAMSHIFT	96.00%	-
Rekha et al. [21]	ASL	E,G,H,J,K,N, O,P,Q,S,T,Z	10	DIP	KNN-SVM/HMM	90.0%	96.00%
Patil et al. [20]	ASL	-	-	glove	ad-hoc	not mentioned	-
Elmahgiubi et al. [11]	ASL	E,M,N,S,T,Z	-	glove	ad-hoc	96.00%	-
Mehdi et al. [15]	ASL	J,Z	-	glove	ANN	88.00%	-
Vamplew et al. [22]	Australian	N/A	N/A	glove	ANN	94%	-

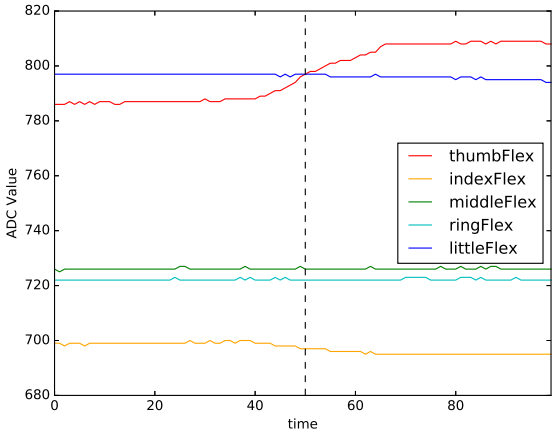


Figure 17: Flex sensor values corresponding to gestures for 'I', gradually followed by deformed gestures.

Among the works based on DIP, the work by Kuznetsova et al. [13] showed the best accuracy of 97.55%. However, they left out the two dynamic characters of ASL, 'J' and 'Z'. Nabiyev et al. [16] showed an accuracy of 96% on only the six vowels of the Turkish alphabet. Although some works based on DIP included the dynamic characters [9, 18, 19], no clarification was provided regarding how the recognition of dynamic characters was achieved. Liwicki et al. [14] worked on double-handed fingerspelling in BSL. They used HMM for sign language modeling. The accuracies for single character recognition and fingerspelt word recognition were 84.1% and 98.9% respectively. However, the latter accuracy was obtained when the HMM was trained using 100 words. If more words are added, the accuracy starts to fall. Rekha et al. [21] used a similar approach, although the domain of their work was much smaller. They covered 15 letters of ASL and 10 words formed by these letters.

Among the works based on data gloves, the reported best accuracy of 96% was obtained by Elmahgiubi et al. [11]. However, there are a few drawbacks of this work. Firstly, they left out six letters from the ASL alphabet including one dynamic letter. Although they covered the other dynamic letter 'J', they did not clarify the mechanism behind this. Secondly, they performed classification using a range-based query without clarifying how they came up with the ranges. Finally, sign language modeling was not covered in their work. Patil et al. [20] claimed to classify all the 26 letters of ASL using flex sensors only. Flex sensors can only detect bending, they did not clarify how they detected contact, orientation, and motion using only flex sensors. They did not provide any accuracy of their system, and just like the previous work, left out sign language modeling. Mehdi et al. [15] achieved 88% accuracy over ASL alphabet using a commercially available glove from 5DT [3]. They left out the dynamic letters and sign language modeling just like the previous works. Vamplew et al. [22] worked on 52 gestures (including dynamic symbols) of Australian Sign Language (Auslan) using a commercially available glove from CyberGlove [2] and achieved an accuracy of 94%. However, sign language modeling was not discussed by them either.

6 CONCLUSION

This work presents ASL and BdSL alphabet recognition using data gloves. Recognition of both dynamic and static character of ASL and BdSL is possible using the developed system. Sign language modeling has also been discussed in this work. Although sign language modeling using image processing technique has received some attention from the research community, not much work has been done for sign language modeling using data gloves. From that perspective, this work provides a new insight into this problem. The lack of dataset creates hindrance in the analysis of data glove based systems. This work attempts to alleviate this problem by generating a dataset containing 1000 data points for each of the letters of ASL and BdSL.

In future, the work can be extended by improving the sign language modeling. Although the current solution works at an acceptable level, it is still prone to noise. The research community has been using Hidden Markov Models (HMM) in speech recognition for quite some time now. Following this direction, researchers have attempted to use HMM in sign language recognition using image processing. Perhaps the same direction can be taken and HMM can be employed for sign language recognition using data gloves. Apart from HMM, the application of LSTM can also be studied.

Currently, if someone wants to deliberately mislead the system, it is possible to do so by giving signs which are not exactly the same as valid signs but are slight derivatives. Future research can be conducted to recognize these aberrant gestures. Moreover, a single glove might not be suitable for all types of sign languages. A unified design could be constructed for a data glove which would be suitable for most, if not all, of the different sign languages.

The system makes predictions in discrete time intervals, without taking into consideration time series properties. Time series analysis might be done for continuous assessment of symbols. The dataset does not contain dynamic gestures, the dataset could be enriched by collecting time series data of these gestures, along with fingerspelling data of words or sentences.

ACKNOWLEDGMENTS

The work was partially supported by the research grant under Higher Education Quality Enhancement Project (HEQEP) CP-3137 awarded by World Bank and approved by University Grant Commission (UGC), Bangladesh.

REFERENCES

[1] [n. d.]. *ASL – American Sign Language.* Available online at http://lifeprint.com/asl101/fingerspelling/index.htm. Last accessed 12:37 am, January 12, 2018.
[2] [n. d.]. *CyberGlove Systems LLC.* Available online at http://www.cyberglovesystems.com/. Last accessed 8:49 pm, August 05, 2017.
[3] [n. d.]. *Data Gloves | 5DT.* Available online at http://www.5dt.com/data-gloves/. Last accessed 8:51 pm, August 05, 2017.
[4] [n. d.]. *How Much Do Sign Language Interpreters Get Paid?* Available online at work.chron.com/much-sign-language-interpreters-paid-8154.html. Last accessed 9:35 pm, August 04, 2017.
[5] [n. d.]. *Sign Language: Fingerspelling.* Available online at http://www.lifeprint.com/asl101/pages-layout/fingerspelling.htm. Last accessed 6:32 pm, April 24, 2017.
[6] [n. d.]. *Sign Language Recognition using Data Gloves.* Available online at https://saquib2527.github.io/slr.html#dataset. Last accessed 3:24 pm, December 19, 2018.
[7] [n. d.]. *WHO | Deafness and hearing loss.* Available online at http://www.who.int/mediacentre/factsheets/fs300/en/. Last accessed 9:08 pm, August 04, 2017.
[8] 1994. *Bengali Sign Language Dictionary.* National Center for Special Education.
[9] Rini Akmeliawati, Melanie Po-Leen Ooi, and Ye Chow Kuang. 2007. Real-time Malaysian sign language translation using colour segmentation and neural network. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference (IMTC), Warsaw, Poland, May 1-3.*
[10] Cao Dong, Ming C Leu, and Zhaozheng Yin. 2015. American sign language alphabet recognition using microsoft kinect. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.*
[11] Mohammed Elmahgiubi, Mohamed Ennajar, Nabil Drawil, and Mohamed Samir Elbuni. 2015. Sign language translator and gesture recognition. In *Proceedings of the IEEE Global Summit on Computer & Information Technology (GSCIT), Sousse, Tunisia, June 11-13.*
[12] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227* (2014).
[13] Alina Kuznetsova, Laura Leal-Taixé, and Bodo Rosenhahn. 2013. Real-time sign language recognition using a consumer depth camera. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV), Sydney, Australia, December 1-2.*
[14] Stephan Liwicki and Mark Everingham. 2009. Automatic recognition of fingerspelled words in british sign language. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on.*
[15] Syed Atif Mehdi and Yasir Niaz Khan. 2002. Sign language recognition using sensor gloves. In *Proceedings of the IEEE International Conference on Neural Information Processing (ICONIP), Orchid Country Club, Singapore, November 18-22.*
[16] VV Nabiyev and S Bayrak. 2006. An artificial neural network approach for sign language vowels recognition. In *Proceedings of the International Scientific Conference of Problems of Cybernetics and Informatics (PCI), Baku, Azerbaijan, October 24-26.*
[17] Joe Naoum-Sawaya, Mazen Slim, Sami Khawam, and Mohamad Adnan Al-Alaoui. 2006. Dynamic system design for american sign language recognition. In *Proceedings of ISCCSP.*
[18] Sabaheta ogi and Gunay Karli. 2014. Sign Language Recognition using Neural Networks. *TEM Journal* 3, 4 (2014), 296–301.
[19] Pratibha Pandey and Vinay Jain. 2015. An efficient algorithm for sign language recognition. *computer* 6 (2015), 7.
[20] Kiratey Patil, Gayatri Pendharkar, GN Gaikwad, and Savitribai Phule. 2014. American sign language detection. *International Journal of Scientific and Research Publications* 4, 11 (2014).
[21] J Rekha, J Bhattacharya, and S Majumder. 2011. Hand gesture recognition for sign language: A new hybrid approach. In *The 2011 International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV).*
[22] Peter Vamplew and Anthony Adams. 1996. Recognition of sign language gestures using neural networks. In *Proceedings of the European Conference on Disabilities, Virtual Reality and Associated Technologies (ECDVRAT), Maidenhead, UK, July 8-10.*
[23] Tim Van den Bulcke, Koenraad Van Leemput, Bart Naudts, Piet van Remortel, Hongwu Ma, Alain Verschoren, Bart De Moor, and Kathleen Marchal. 2006. SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC bioinformatics* 7, 1 (2006), 43.
[24] Tamás Varga and Horst Bunke. 2003. Generation of synthetic training data for an HMM-based handwriting recognition system. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on.* IEEE, 618–622.