DyFAV: Dynamic Feature Selection and Voting for real-time recognition of fingerspelled alphabet using wearables

Prajwal Paudyal, Junghyo Lee, Ayan Banerjee, and Sandeep K.S. Gupta
IMPACT Lab, http://impact.asu.edu
Arizona State University, Tempe, Arizona
{ppaudyal, jlee375, abanerj3, sandeep.gupta}@asu.edu

ABSTRACT
Recent research has shown that reliable recognition of sign language words and phrases using user-friendly and non-invasive armbands is feasible and desirable. This work provides an analysis and implementation of including fingerspelling recognition (FR) in such systems, which is a much harder problem due to lack of distinctive hand movements. A novel algorithm called DyFAV (Dynamic Feature Selection and Voting) is proposed for this purpose that exploits the fact that fingerspelling has a finite corpus (26 letters for ASL). The system uses an independent multiple agent voting approach to identify letters with high accuracy. The independent voting of the agents ensures that the algorithm is highly parallelizable and thus recognition times can be kept low to suit real-time mobile applications. The results are demonstrated on the entire ASL alphabet corpus for 9 people with limited training and average recognition accuracy of 95.36% is achieved which is better than state-of-art for armband sensors. The mobile, non-invasive, and real time nature of the technology is demonstrated by evaluating performance on various types of android phones and remote server configurations.

Author Keywords
Sign language processing; gesture-based interfaces; assistive technology; wearable and pervasive computing.

ACM Classification Keywords
K.4.2 Social Issues: Assistive technologies for persons with disabilities; H.5.2 User Interfaces: User-centered design

INTRODUCTION
The advent of wearable electronics has opened up a new era of pervasive human-centered computing which when applied to disability research results in technology of significant impact. Sign language recognition (SLR) systems is such an example, where wearable technology can be used to drastically improve communication between the sign language cognizant deaf and hard of hearing and the agnostic mass. These SLRs collect data from wearables to recognize gestures, which can be then converted into text or speech. In this paper, we focus on American Sign Language Recognition (ASLR) systems and propose a novel mobile finger-spelling recognition system using wearable wristbands.

American Sign Language (ASL) (potentially true for many other SLs) has four important aspects (Figure 1): a) word gestures for commonly used words, b) finger-spelling, through which unique hand gestures are associated to a letter in the English (or other language) alphabet [34], c) emotion or expression, through which abstract concepts such as intensity and finer expressions are depicted, and d) individual or customized variations. Gesture recognition research so far has mostly focused on the first two aspects of ASL. Although there are several non-invasive wearable systems available for word recognition, accurate finger-spelling recognition is still a challenge. This is due to the fact that ASL signs for words use significant arm movements that can be easily captured using sensors such as accelerometers, or gyroscope available in state-of-the-art wearables. However in finger-spelling, arm movements are restricted, rather relative positioning of the fingers are important distinguishing factors amongst gestures for different words. Such nuances are difficult to capture using wearables and may require other infrastructure such as video monitoring [4]. Electromyogram (EMG) signals, which measure the tension in muscles are available in state-of-the-art wristbands, which can be potentially used for recognizing relative finger positioning. However, EMG sensors have very low signal to noise ratio, and vary not only from one individual to other but also during different attempts by the same individual.

In this paper, a novel algorithm called DyFAV (Dynamic Feature Selection and Voting) is proposed that exploits the fact that fingerspelling has a finite corpus (26 letters for ASL) and uses an independent multiple agent voting approach to identify letters with high accuracy. The custom approach is shown to outperform other but also during different attempts by the same individual.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
IUI 2017, March 13-16, 2017, Limassol, Cyprus
© 2017 ACM. ISBN 978-1-4503-4348-0/17/03... $15.00
DOI: http://dx.doi.org/10.1145/3025171.3025216

Need for fingerspelling
An ASL user would use the American Fingerspelled Alphabet (AFA), (also called the American Manual Alphabet). The AFA

Author Keywords
Sign language processing; gesture-based interfaces; assistive technology; wearable and pervasive computing.

ACM Classification Keywords
K.4.2 Social Issues: Assistive technologies for persons with disabilities; H.5.2 User Interfaces: User-centered design

INTRODUCTION
The advent of wearable electronics has opened up a new era of pervasive human-centered computing which when applied to disability research results in technology of significant impact. Sign language recognition (SLR) systems is such an example, where wearable technology can be used to drastically improve communication between the sign language cognizant deaf and hard of hearing and the agnostic mass. These SLRs collect data from wearables to recognize gestures, which can be then converted into text or speech. In this paper, we focus on American Sign Language Recognition (ASLR) systems and propose a novel mobile finger-spelling recognition system using wearable wristbands.

American Sign Language (ASL) (potentially true for many other SLs) has four important aspects (Figure 1): a) word gestures for commonly used words, b) finger-spelling, through which unique hand gestures are associated to a letter in the English (or other language) alphabet [34], c) emotion or expression, through which abstract concepts such as intensity and finer expressions are depicted, and d) individual or customized variations. Gesture recognition research so far has mostly focused on the first two aspects of ASL. Although there are several non-invasive wearable systems available for word recognition, accurate finger-spelling recognition is still a challenge. This is due to the fact that ASL signs for words use significant arm movements that can be easily captured using sensors such as accelerometers, or gyroscope available in state-of-the-art wearables. However in finger-spelling, arm movements are restricted, rather relative positioning of the fingers are important distinguishing factors amongst gestures for different words. Such nuances are difficult to capture using wearables and may require other infrastructure such as video monitoring [4]. Electromyogram (EMG) signals, which measure the tension in muscles are available in state-of-the-art wristbands, which can be potentially used for recognizing relative finger positioning. However, EMG sensors have very low signal to noise ratio, and vary not only from one individual to other but also during different attempts by the same individual.

In this paper, a novel algorithm called DyFAV (Dynamic Feature Selection and Voting) is proposed that exploits the fact that fingerspelling has a finite corpus (26 letters for ASL) and uses an independent multiple agent voting approach to identify letters with high accuracy. The custom approach is shown to outperform other but also during different attempts by the same individual.
Figure 1: Modules of American Sign Language Recognition

American Sign Language Recognition (ASLR)
- Objective: Seamless gesture communication
- Input: Some form of ASL gestures
- Four core modules to achieve seamless ASLR

Words Recognition (Module 1)
- Objective: Recognize gestures in word level
- Input: Gestures data from sensors
- Output: Word with highest score

Finger Spelling Recognition (Module 2)
- Objective: Recognizing gestures in alphabet level
- Input: Gestures data from sensors
- Output: Letter with highest score

Emotion Expression Recognition (Module 3)
- Objective: Recognizing emotion
- Input: Facial images from camera sensor
- Output: Emotions as relevant to ASL

Customized Gesture Recognition (Module 4)
- Objective: Recognizing gestures for user-defined words
- Input: Gestures data from sensors
- Output: Word with highest score in private database

consists of 22 handshapes (1) that when held in certain positions and/or are produced with certain movements represent the 26 letters of the American alphabet [19]. There has been some work recently [24, 39, 14] that use armbands or other sensors on the arm to get information about sign language gestures, however there is not any work that aims to incorporate fingerspelling along with it.

The use of fingerspelling in ASL has been studied extensively and researchers agree that fingerspelling is integrated into ASL in very systematic ways [5, 23]. The most obvious usage is primarily for representing proper nouns or for English words without equivalents in sign language called neutral fingerspelling [3, 37, 22]. Beside that, Lexicalized fingerspelling, abbreviations for longer signs, two-word compounds, initialized signs and singed-fingerspelled compounds [2] all utilize fingerspelling. Fingerspelling also helps to bridge the gap created due to the variation of ASL lexicon across geography and cultures [6]. This tight integration with ASL means that an ASLR system that aims to translate conversations in ASL to English must incorporate fingerspelling recognition.

An ASLR system should be real-time, non-invasive and pervasive. Fingerspelling has been mostly tackled using image, video, depth camera, data gloves. Some work (summarized in Table 2) do use sensors on the arm and armbands as discussed in section Related Work, however, their accuracy is around 75%. Two inferences can be made from Table 2: 1. Finger-spelling recognition has been restricted mostly to image/video or data-glove based approaches 2. The approaches that do use arm bands or sensors on the arm either do not support all the letters, are not real-time or are too invasive for daily usage. This work functions to bridge this gap.

Fingerspelling recognition is a very different problem than word recognition and has the following challenges:

Diversity of significant features: Each sign in the AFA utilizes a unique hand configuration. While letters like ‘J’ and ‘Z’ show considerable hand movements and others like ‘Q’ and ‘R’ have some amount of orientation changes, this is not true for all letters of the alphabet. This means that there usually isn’t enough information that can be collected from the accelerometer and gyroscope sensors alone that can help classify
between the 26 classes. Hence, a recognition algorithm that focuses only on one modality such as accelerometer, gyroscope, orientation or EMG will have low accuracy [1]. Although the overall structure for signing a letter is specified (Table 1), each person has subtle nuances in correctly executing the sign. Hence, an AFA recognition system should not only be able to fuse multiple modalities but also learn the unique signing patterns of an individual much like handwriting recognition does.

In this paper, we propose Dynamic Feature selection And Voting (DyFAV) algorithm that extracts significant features from accelerometer, gyroscope, orientation and EMG signals and assigns a weight for each of these features such that it is fine-tuned not to the specific patterns that help distinguish between letters, but also to specific nuances in how an individual signs them.

**Lack of significant arm movement:**

There is a considerable lack of arm movement while performing most of the AFA signs. Thus, classification has to depend on hand configurations, which can be captured using EMG data [1] Some letters of the ASL alphabet are very closely related to others and differ only very slightly in finger placement. For instance the letters ‘M’ and ‘N’ only differ from each other by whether the thumb is in between the first and second or the second and third fingers (Figure 2).

Further, EMG data can vary a lot between people [15] and feature-selection on this data is a difficult problem since different hand configurations activate different portions of muscles in varying ways. Hence, an AFA recognition system should be able to pick up fine differences at the different portions of the arm to distinguish between these closely related signs. To account for this, the voting mechanism of DyFAV ranks the features such that the ones that were most instrumental in distinguishing a particular sign during training get the highest weights.

**DyFAV approach**

A notable difference between recognizing words versus letters is that fingerspelling has a finite corpus i.e. for ASL we have only 26 possible classes. DyFAV takes advantage of this to select a dynamic list of salient features for each letter and lets each of those salient features vote. These votes are adjusted by a dynamic weight for each feature that is learned during training. Details on the DyFAV algorithm can be found in Section Methodology and the recognition accuracy and time analysis can be found in Section Evaluation.

A study that was performed using traditional machine learning approaches of SVM, Naive Bayes, Random Forest and MLP gave respective average accuracies of 81.38 %, 90.66%, 80.47% and 90.45 %. DyFAV on the other had gave us an average accuracy of 95.36 %.

The contributions of this study can be summarized as the following: (1) DyFAV algorithm which uses dynamic feature selection and voting to solve a fixed class classification problem (2) Highly efficient recognition algorithm capable of real-time recognition of fingerspelling using a smartphone (3) A mobile app user-interface that allows training, testing, user driven correction and collection of user feedback (4) Evaluation of the accuracy and execution time on 9 users using a computer and different types of smartphones.

**RELATED WORK**

The motivation behind this work is to facilitate conversation by providing a pervasive and practical way to recognize fingerspelled letters using a commodity sensor and limited training. We evaluate the existing work in this field by dividing them first by the type of input devices used and then further by their invasiveness, mobility, real-time support, accuracy and degree of support for word/fingerspelling recognition (Table 2).

**Data glove based approaches**

Data glove based approaches have been used extensively to recognize sign language alphabet as well as words [10, 11, 13, 32, 18, 9]. Due to the presence of finger joint information, it becomes relatively easy to classify the various finger configurations. Contrasting to the image and video based approaches, the data glove based approaches don’t suffer from problems of occlusion, lighting changes and background variations. However, these approaches have been found to be invasive as wearing a data glove prevents the user from participating in day-to-day activities as well as singles the sign language user out. The proposed work in this paper attempts to solve this problem by providing an alternative that uses inconspicuous armbands (Myo) to gather the information.

**Image, video and depth based approaches**

Fingerspelling forms distinct shapes thus it is very intuitive to utilize image and video based approaches to classify the vari-
Table 1: ASL symbols and their hand configurations

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat hand</td>
<td>A, S, T</td>
<td>The hand clasped with thumb</td>
</tr>
<tr>
<td>Flat hand</td>
<td>B</td>
<td>The open or spread hand, thumb out or in</td>
</tr>
<tr>
<td>Curved hand</td>
<td>C, O</td>
<td>Fingers connected with thumb or not</td>
</tr>
<tr>
<td>Retracted</td>
<td>E</td>
<td>The fingers clenched to palm</td>
</tr>
<tr>
<td>F-hand</td>
<td>F</td>
<td>Thumb and forefinger touch, other fingers spread</td>
</tr>
<tr>
<td>Index</td>
<td>G</td>
<td>Allocheric forms: g, d, l</td>
</tr>
<tr>
<td>Y-hand</td>
<td>H</td>
<td>First two fingers extended and joined</td>
</tr>
<tr>
<td>Pinky</td>
<td>I</td>
<td>The little finger projects from the closed hand</td>
</tr>
<tr>
<td>K-hand</td>
<td>K</td>
<td>The index, second and thumb make k</td>
</tr>
<tr>
<td>L-hand</td>
<td>L</td>
<td>The thumb and index make right angle</td>
</tr>
<tr>
<td>Bent hand</td>
<td>M</td>
<td>The hand makes a dihedral angle</td>
</tr>
<tr>
<td>R-hand</td>
<td>R</td>
<td>The first two fingers crossed, r</td>
</tr>
<tr>
<td>V-hand</td>
<td>V</td>
<td>The index and second extended and spread</td>
</tr>
<tr>
<td>W-hand</td>
<td>W</td>
<td>The first three fingers extended and spread</td>
</tr>
<tr>
<td>Y-hand</td>
<td>Y</td>
<td>The thumb and little finger are spread out from</td>
</tr>
</tbody>
</table>

Figure 4: Left: Recognition for ‘P’; Right: Testing for ‘Y’.

Various hand configurations that relate to the various letters (and numbers). Image and Video based approaches have achieved high accuracy rates [31, 27, 25, 33]. There have been some approaches of classifying continuous gestures using video with high accuracy [35]. Furthermore, there have been approaches that use the depth information provided by the Kinect sensor to improve classification accuracy. Image and video based approaches need special equipment setup prior to usage which is a major limitation. In addition they face the unavoidable problems of occlusion, sensitivity to ambient light and background changes. Furthermore, classification between visually similar letters like ‘S’, ‘M’ or ‘N’ is difficult with images. To get around this problem, some researchers have evaluated their systems on on subsets of the alphabet [27]. These limitations prevent image, video and depth based systems from being practical for day-to-day usage.

Sensors on the arm based approaches

Using armband or sensors on the arm is a relatively new and promising approach to SLR that utilizes various combinations of sensors like accelerometer, gyroscope and EMG [24, 29, 38]. Some other works have focused on providing algorithms for gesture recognition for control of devices [20]. The accelerometer and gyroscope sensors provide location and movement information while the EMG sensors help to classify the finger configurations. Recognizing fingerspelling, however, provides a unique challenge since there is little of no movement involved in most of the letters in ASL and EMG signals can be noisy. Savur et al. have proposed a real-time and an offline system to translate ASL fingerspelling to letters. The authors report an accuracy of 82.3 % on a real-time system and a 91.1 % on an offline system using all 26 letters repeated 20 times each for training [29]. The approach we discuss in this paper achieves a higher real-time average accuracy of 95.36%. Also only 5 training instances are required which is much smaller than other works [30, 27]. Further, we use commodity wireless sensors in place of the medical grade sensors, and perform all testing and training using a smartphone app (to be available in play store). To increase accuracy we utilize accelerometer, orientation and gyroscope sensors in addition to the EMG sensors.

PROBLEM DESCRIPTION

Input: Discrete signals for accelerometer, gyroscope, orientation and EMG from Myo Devices.
Output: Recognized alphabet.
Challenges:
1. Low SNR signal: The signals, specially the EMG have a very low signal to noise ratio. Thus, it makes it difficult to design an algorithm that performs consistently.
2. Difference between people: The signals, specially the EMG varies considerably between people even when performing the same alphabet
3. Diversity of significant features: A different combination of significant features appears that helps to identify different alphabets.
4. Lack of significant movement: Fingerspelling as opposed to other ASL gestures do not involved much arm movement

Constraints:
1. Real-time constraint: To facilitate recognition in real-time, the recognition algorithm must be light-weight.
2. Limited processing: Recognition is done in the smartphone to remove network dependence and lag. Smartphones don’t have as much processing power as servers.
3. Limited Training Data: In order to make the usability of the system high, training sessions are repeated only 5 times per letter per person. This puts a constraint on the amount of data available for training.

The core problem we solve in this research is a fixed-class classification problem of identifying a letter performed using just the signals captured by a wearable armband. The signals captured are noisy and fluctuate between people. Some of the features extracted are critical in helping for some classes, but not so much for others. For instance, features from gyroscope and orientation sensors are critical for letters ‘J’ and ‘Z’ but they are not so useful for other letters.

In addition, the recognition has to be done under many constraints. The system requires a very small number of training instances(5) which contributes to high usability. However, this also means that we only have limited instances of data to learn from with 510 features each. Thus, using out of the box machine learning techniques becomes less feasible. For the same reasons, common dimensionality reduction techniques like Principal Component Analysis (PCA) are not very helpful. Careful selection of features has to be done by relying heavily on the differences among the various signs and how people use them.

The aim of facilitating real-time usage puts a constraint on the complexity of the recognition algorithm and justifies the
Table 2: Existing work in sign language recognition.

<table>
<thead>
<tr>
<th>Input device</th>
<th>Work</th>
<th>Reported Accuracy</th>
<th>Word Lexicon</th>
<th>FR support</th>
<th>Real-Time</th>
<th>Invasiveness</th>
<th>Continuous</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Glove</td>
<td>[12] (2014)</td>
<td>up to 94%</td>
<td>NA</td>
<td>5 of 26</td>
<td>NA</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[27] (2015)</td>
<td>up to 95%</td>
<td>NA</td>
<td>6 of 26</td>
<td>NA</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[19] (2014)</td>
<td>94%</td>
<td>NA</td>
<td>26 of 26</td>
<td>NA</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[21] (2015)</td>
<td>81% - 94%</td>
<td>300</td>
<td>26 of 26</td>
<td>NA</td>
<td>Medium</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Image/Video/Depth Sensors on the arm</td>
<td>[31] (2015)</td>
<td>82%</td>
<td>300</td>
<td>NA</td>
<td>No</td>
<td>High</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[27] (2015)</td>
<td>69% - 75%</td>
<td>NA</td>
<td>26 of 26</td>
<td>Yes</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[10] (2015)</td>
<td>85% - 97.8%</td>
<td>NA</td>
<td>23 of 26</td>
<td>No</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[25] (2014)</td>
<td>99.04%</td>
<td>NA</td>
<td>26 of 26</td>
<td>No</td>
<td>High</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[30] (2015)</td>
<td>82.3%</td>
<td>20</td>
<td>NA</td>
<td>Yes</td>
<td>Low</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[24] (2016)</td>
<td>91.7%</td>
<td>20</td>
<td>NA</td>
<td>Yes</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>[11] (2016)</td>
<td>41.93%</td>
<td>NA</td>
<td>20 of 26</td>
<td>No</td>
<td>Low</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>This Work (2016)</td>
<td>Av. of 95.36%</td>
<td>NA</td>
<td>26 of 26</td>
<td>Yes</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

All data collected during training is stored in the smartphone and sent to the server via socket connection. The server performs preprocessing, segmentation, feature extraction, feature engineering and assigns weights to features based on how useful they were in classifying the letters in the training set. Each letter receives a different set of features and each feature in turn receives a different weight for each letter. Details on the algorithm are explained in section Methodology. A single file with all relevant information for all of the alphabet agents is sent back to the phone to be used during recognition as seen in Figure 5. This is the trained model for the user. The training was done with 9 users (5 females and 4 males) between the ages of 20 and 35 who were taught to sign the letters as part of the experiments.

Recognition
During usage, the test module of the app is brought up to begin translation. Data is collected for 5 s. for each letter and recognition is done in the phone. The trained model that is obtained from the server is saved locally and used by the 26 alphabet agents to get votes from the various top features as explained in Methodology. These votes are added up to compose the total score and the alphabet with the maximum score is shown as the correct alphabet. This is summarized in Figure 6. A simple wave out option is provided for the user. The recognized alphabet is then shown on the screen and spoken out (Figure 4). In case of misclassification, the user can offer a correction. This feedback data is recorded and sent to the server for improvements. This correction and feedback feature is optional and can be turned off. For a user who is learning to sign this module can be used as practice.

An end-to-end architecture for offloading the decision is also provided as suggested by [26]. The recognition on the server is very fast as can be seen in Figure 9. However, there is the added time due to network latency. The decision to offload the computation to the server depends on availability of network, speed of network, processing power of the smartphone, battery level as well as user preferences.

After recognition, a text-to-speech library is utilized to speak out the letter to facilitate conversation with hearing users. The recognized letter can also be utilized by an application that supports input by dictation.

USAGE AND SYSTEM ARCHITECTURE
Training
During training time, the user performs each of the 26 alphabet a total of 5 times while wearing a Myo armband as shown in Figure 4. The average total training time for the entire corpus across 9 users was about 10 minutes. While the entire training session could easily be completed in a single session without much fatigue, for best results training should be done in multiple sessions to capture some variations. The training screen (Figure 3) has an indicator of connection to the sensor, a progress bar, and a dropdown to select the letter and a guidance animation to assist in training.
METHODOLOGY

Training

Data Collection

The raw data for all signals is obtained by using the ‘Myolib’ Library by darken [8]. Orientation values are received in the form of unit quaternions. The pitch, yaw and roll values are obtained from the quaternion values w, x, y and z by using the following equations:

\[ \text{roll} = \tan^{-1}\left(\frac{2(wx + yz)}{1 - w^2 - x^2 - y^2}\right) \]

\[ \text{pitch} = \sin^{-1}(\text{max}(-1, \text{min}(1, 2(wy - xz)))) \]

\[ \text{yaw} = \tan^{-1}\left(\frac{2(wx + yz)}{1 - w^2 - x^2 - y^2}\right). \]

Zeroing and Normalization
To account for different starting positions, orientations and rest-level activations of EMG sensors, all sensor readings are zeroed at initialization point. This is done by differencing the average of the first three sensor readings from the rest of the signal. Three readings are used in place of one to minimize noise. In addition, normalization is performed across the entire alphabet corpus and abnormal sensor readings are smoothed. Normalization does not play a role in increasing the accuracy for an individual but does have an impact if it is used with data from other people.

Algorithm 1 Feature Engineering

```
1: procedure SORT BY FEATURES(F_D)
2: for (i in 1 to 26) do
3: for (j in 1 to number_of_features) do
4: sort(F_D.feature[j])
5: range[j] ← range(letters[j].feature_data)
6: thres_lower[j] ← range[0]
7: thres_upper[j] ← range[1]
8: weight[j] ← abs(ln(diff(range) − 4)/130 − 4)
9: end for
10: n_min ← min(weight)
11: n_max ← max(weight)
12: norm_weight ← (weight − n_min)/(n_max − n_min)
13: write_feature_file(feature.range.norm_weight)
14: end for
15: end procedure
```

Feature Selection
For each one of the channels we select five features: 1. Max 2. Min 3. Mean 4. Standard Deviation and 5. Total Energy. Features were chosen to incorporate the variation in data while keeping computation to obtain the features minimal. The EMG has 8 channels and each of the other signals have 3 channels. This gives us a total of 85 features. We then compute the same features, but after segmenting the data into 5 segments. We do this to preserve some time domain information after feature selection as suggested by [36]. Thus, we end up with a total of 510 features for each instance.

Feature Engineering
Feature data from all instances in the previous step is loaded in a matrix. This gives us a 130 X 510 matrix F_D. This matrix is sorted in ascending order based on the values of each of the 510 features. Each feature then receives a weight based on how well it is able to perform in sorting each letter in the vicinity of other instances of the same letter. This process is performed iteratively for each feature to get a different weight \( \text{weight}(j) \) for each feature for each letter. After completing the entire process, all the weights are normalized. These normalized weights are used during recognition as voting factors (Algorithm 1). After this step, the lower and upper thresholds corresponding to each feature is expanded so that values that test features that lie sufficiently close to the actual threshold are still counted towards final voting. This is done to prevent over-fitting to training data. The output of this step is the ‘Feature Model’ as seen in Table 3. It can be seen that the mean standard deviation across the training instance for EMG pod 2 (EMG2_STD0) was the most instrumental feature that could contain all five training instances within a range of 10, thus this particular feature has one of the highest voting weights.

\[ \text{adj.thres.l} = \text{thres.lower} - (\text{thres.range}/\text{range}) \]

Recognition and User Interface
The data collected during testing goes through similar steps of preprocessing and feature selection. These steps were intentionally kept simple at the training phase, thus there’s a lot of computational load at this time. Then, the various recognition agents are called that perform a feature based voting method and output a single probability score for the likelihood of that set of features belonging to any particular letter (Figure 6). These ‘agents’ do the testing based on the normalized feature weight (\( \text{norm.weight} \)) they have for each feature from the model (Model_A) generated at the training phase. Each of the feature is allowed to cast a vote only if it is able contain the test feature within its adjusted lower threshold (A_lower) and adjusted upper threshold (A_upper). The vote that is cast by each feature is equivalent to the normalized weight corresponding to it in the trained feature model. Since the weight decreases logarithmically with increase in range of training classification, the vote of a feature that has a wide range at training is heavily discounted even though it could successfully classify the test feature. This ensures that the features that do well during training are counted the heaviest. On the other hand, if a feature that performed well during training is not able to contain a test feature, it is ignored for recognition. This insures the recognition step is more robust to overfitting. The voting algorithm for each server agent is shown in Algorithm 2. Since each ‘alphabet agent’ works with a unique set of data for that alphabet, the recognition is executed in parallel before getting the final votes for comparison.

Table 3: Top 10 Features in model for Alphabet ‘A’.

<table>
<thead>
<tr>
<th>Features</th>
<th>Range lower</th>
<th>Range upper</th>
<th># Lower</th>
<th># Upper</th>
<th>Norm Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMG1 STD0</td>
<td>117</td>
<td>124</td>
<td>119</td>
<td>124</td>
<td>0.4815</td>
</tr>
<tr>
<td>EMG2 STD0</td>
<td>114</td>
<td>124</td>
<td>119</td>
<td>123</td>
<td>0.4815</td>
</tr>
<tr>
<td>EMG7 TOTAL ENG0</td>
<td>114</td>
<td>124</td>
<td>119</td>
<td>123</td>
<td>0.4815</td>
</tr>
<tr>
<td>ORN1 MEAN4</td>
<td>20</td>
<td>30</td>
<td>21</td>
<td>30</td>
<td>0.4815</td>
</tr>
<tr>
<td>ORN1 MEAN5</td>
<td>19</td>
<td>29</td>
<td>19</td>
<td>29</td>
<td>0.4815</td>
</tr>
<tr>
<td>ORN1 TOTAL ENG0</td>
<td>19</td>
<td>29</td>
<td>19</td>
<td>29</td>
<td>0.4815</td>
</tr>
</tbody>
</table>
Algorithm 2 Test Agent

1: procedure ISALPHABET_A(test_features)
2:  sum ← 0
3:  Model_A ← read_model_A from training
4:  this_feature ← Model_A[feature]
5:  A_lower ← Model_A[thres_lower adj]
6:  A_upper ← Model_A[thres_upper adj]
7:  norm_weight ← Model_A[norm_weight]
8:  for (this_feature in trained_model_A) do
9:    if this_feature in range(A_lower, A_upper) then
10:       sum ← sum + norm_weight
11:  end if
12: end for
13: return sum

Flexible approach and User Feedback
A flexible approach that is able to optimize the number of features used for recognition according to the processing power of the smartphone is utilized. In other words, a trade-off is performed between accuracy and performance to meet real-time constraints. For a Nexus 5 smartphone, the average accuracy when keeping the processing time under .5 s was found to be 81.65% using 64 features and 1 s was found to be 89.22% using 130 features (Figure 9). The highest average accuracy that was obtained was 95.36% and used 327 features. The recognition time for this number of features on the Nexus 5 smartphone was just over 2 s. However when we used a smartphone with a more powerful processor, the recognition time was cut in half. Thus, the recognition accuracy automatically adjusts with available resources while maintaining a desired recognition time.

To account for the loss in accuracy due to timing constraints and slower processing speeds, and to obtain valuable feedback from the user, we implemented a simple User Interface that allowed users to toggle to the alphabet identified next in line according to the final score. Figure 11 shows how accuracy increases as more letters are returned. When only 60 features were used for recognition in the slower Nexus 5 phone, for a recognition time under .5 s, the average chance of having the correct alphabet within the first two choices goes to 92.2% while having it within the first five choices goes up to 98.5%. This shows that having this simple user interface that allows the user to quickly select between top-ranked choices greatly increases accuracy and makes the system usable even with strict constraints (detailed analysis in Section Evaluation).

RESULTS AND EVALUATION
The most important metrics that the system is tested against are recognition speed, usability and accuracy. Experiments were designed with these metrics in mind.

Design of Experiments
Training data was collected in 3 sessions. In each session, the users performed the entire alphabet corpus (A-Z) twice. All training was done in a lab setting using a smartphone app. Users were seated in a chair and were free to rotate or move their hands. The training was done using the first five samples when the remaining one was used for testing. Results were computed using cross-validation. The smartphone app played a GIF image of the alphabet being signed during training to help keep the signs consistent among the users. Video data was collected and later analyzed to make sure the users were using the signs correctly. The device specifications are as follows: Smartphone 1: Nexus 5: 2.26 GHz quad-core Snapdragon 800 processor Smartphone 2: Qualcomm MSM8974AC Snapdragon 801 One Plus One: Quad-core 2.5 GHz Krait 400 Smartphone 3: Samsung Galaxy Note 5: Exynos 7420 Octa: Quad-core 1.5 GHz Cortex-A53 & Quad-core 2.1 GHz Cortex-A57

Evaluation
To evaluate the effectiveness of the feature ranking algorithms we performed a test on data collected for all 26 letters for 9 people. Figure 8 shows how the accuracy changes as more features are used. The average accuracy across all people is shown in blue. The highest maximum of 100% accuracy across all people is reached using 241 features, the highest average accuracy of 95.36% is reached using 327 features. It can be seen that the maximum accuracy stalls after a certain number of features and sometimes even reduces. This is because some of the lesser important features that aren’t good classifiers slowly contribute to decrease the accuracy. The optimal number of features to be used based on these experiments is determined to be around 327 across users as the accuracy at this number stays the highest.

Figure 9 shows how the recognition time varies with the increasing number of features. The blue line represents the average recognition time across users on the server, which has an i7 processor. Three different Android smartphones as specified in Section Design of Experiments were tested with the algorithm for increments of 10 features. The average total execution time on the server for all 510 features was still under 100 ms. and that for 327 features (optimal number) is 67 ms. However, on the smartphone the execution time for 510 features was over 3 s. for Nexus 5 and One Plus while it was just over 2 s. for a Galaxy Note 5. The Note 5 could reach the optimal average accuracy using 327 features in just under 1 s. while it required close to 2 s. for the Nexus 5 and One Plus phone. Thus, a different optimal number of features should be computed and used for different smartphones based on their computational capabilities while ensuring execution time remains acceptable.

Another way to increase usability of the system when there are strict timing constraints was to offer close matches in ranked order to user. She can then do a wave out gesture in the 1 s. window between gestures to select the next one in line instead. The next in line alphabet would appear from the right of the smartphone in gray color before being brought into focus. To test whether such an approach would increase accuracy we tested the probability of the correct letter being returned in a set of 1, 2, 3, 4 and 5. The maximum average accuracies that were achieved were 95.19 %, 98.82%, 99.41 %, 99.84 % and 99.92% respectively. This is summarized in Figure 11. The biggest jump in accuracy between the return set of 1 vs. 2 justifies to have this feature present even in a very capable smartphone to ensure highest performance. This feature will be especially important in smartphones with lower processing speeds. Using just 100 features, the accuracy at a return set of 2 is over 96% while it is only 87% for a return set of 1. Also
it should be noted that the accuracy for a return set of 4 quickly converges to practically 100 % when number of features is 170. Thus, the return set of 5 is not needed. An aggregated confusion matrix across the 9 users is shown in Figure 7. It can be seen from this confusion matrix that letters like ‘J’ and ‘Z’ have done extremely well due to the inclusion of hand rotation and movement while letters like ‘S’, ‘T’ or ‘N’ are not affected. Intuitively, this makes sense because the letters ‘A’, ‘S’ and ‘O’ are very similar to each other and are thus frequently misclassified.

Figure 10 shows how important the different signals were in classifying the letters. The y-axis lists the number of times features related to each of the four signals of accelerometer, EMG, gyroscope and Orientation appeared in the top 20 features for each of the letters shown. It can be seen that orientation signals are high across the board and accelerometer related features are consistent across all letters but EMG related signals and Gyroscope related signals have more variation. Gyroscope related features are the effective to help classify the letters ‘J’ and ‘Z’ which is expected due to the high amount of movement that is required for them. On the other hand, letters ‘M’, ‘N’, and ‘E’ or ‘A’ have few top features that are gyroscope related, but many more that are EMG related. This again makes intuitive sense since these should be classified more by hand configuration rather than rotation information.

**DISCUSSION**

Although EMG data varies a lot between people, the voting mechanism of DyFAV ranked the features such that the features that were most instrumental in distinguishing a particular sign during training get the highest weights which helped increase recognition. DyFAV algorithm extracts significant features from accelerometer, gyroscope, orientation and EMG signals and assigns a weight for each of these features such that it is fine-tuned not only on the specific ways an individual signs the letters, but also takes into account the specific patterns that help identify one alphabet from the other. Although this work contributes a fast and efficient algorithm to recognize fingerspelling, the need for individualized training although very little, is still a hinderance. The data collection is collected in isolation for letters and is thus not very close to suggested levels of fluent ASL signing speeds [28]. The uses of fingerspelling in isolation might not be very significant as it could be replaced by a smartphone notebook or pen/paper, however in the context of an SLR system as well as a system for ASL based dictation or HCI control, the research becomes highly relevant. Not all features were equally relevant in doing the recognition for all people, thus, custom wearables of the future should focus on creation of wearables that assist most
CONCLUSION AND FUTURE WORK

Conclusion
Fingerspelling is an intrinsic part of sign language and thus systems that aim to be usable in day to day communication must support it. For systems with a limited dictionary of supported words and phrases, fingerspelling recognition can bridge the expressibility gap. Fingerspelling recognition using wearables is a difficult and a unique problem that can benefit from specialized algorithms. We used the proposed algorithm to classify all 26 letters with high accuracy and provide analysis on features used and recognition. The algorithm suggested worked well for this application and we foresee that this can be applied to any other classification problem as well.

Future Work
We envision the following improvements and future work:

Windowing
The focus of this work was to test the effectiveness of the DyFav algorithm, thus we utilized clearly marked end points for collecting data. However, when the system is deployed to be used by the public, windowing approaches should be used to collect data continuously.

Test users
The algorithms were tested in a lab setting with limited number of users. A more thorough study needs to be made that involves fluent users of ASL in a more natural setting. A mobile app for this work will be released in the Google Play store very soon. This will allow us to work with data captured from a larger number of test users as well as get recommendations on usability. Also, the feedback that we receive for incorrect recognitions will help to improve the algorithm and application.

User independence
Although we successfully devised a system that is able to be entirely trained in 10 minutes, we still envision a system that would work without training. We can leverage the data we receive from the online deployment of the app to modeling EMG for a larger user base.

ACKNOWLEDGEMENTS
This project was partly funded by NIH NIBIB (Grant #EB019202) and NSF IIS (Grant #1116385). We thank Paul Quinn, Director of ASL at ASU for his invaluable ASL domain knowledge and feedback.

REFERENCES


