Recognition of Sign Language Gestures

Project Final Report

G. Anurag
Heinz College, Carnegie Mellon University
U.S.A.
agandhi@andrew.cmu.edu

M. Charu
Heinz College, Carnegie Mellon University
U.S.A.
cmulwani@andrew.cmu.edu

ABSTRACT

Hearing and speech impaired people often find it difficult to communicate with people who do not know sign language. Our project aims at helping these people by automating sign language detection using machine learning algorithms. We will primarily work on a multivariate, time-series sensor dataset, collected using a NinTendo PowerGlove. We apply different machine learning techniques not limited to neural networks, SVM and a baseline algorithm to compare and contrast the performance of each of these algorithms and parameter settings on our dataset. We build an ensemble soft voting classifier using well calibrated classifiers trained on different feature spaces. The classifier is able to learn temporal features better than the individual estimators in ensemble. We also perform an analysis by building models with just the right-hand features data, and we were able to achieve comparable performance, which leads us to the conclusion that the right hand is the dominant hand and captures most of the variance.

1 INTRODUCTION

According to WHO [1], over 5% of the world population is hearing impaired, which makes the number about 360 million people. They face challenges everyday which range from communicating with other people, to finding jobs compatible with their mode of communication. There have been several studies to make the process of sign language automated in real time and machine learning algorithms have been integral to improve the accuracy of converting signs to sign language linguistic terms. A lot of work has been done around sign language translation involving input from videos and image data. This technique requires capturing sign movements from individual frames in the videos and extracting temporal information to form features. One such methodology is used by H.Cate et.al. called Sequential Pattern Mining [2]. This technique involves finding temporal patterns in the input dataset which serve as unique identifiers for a particular class.

Different techniques have also been used with inputs being from sensor data for hand orientation, rotation and bend movement. Kadous [3], in his paper, describes the use of meta-features, which are essentially important events in a time series which give more meaning to the original features.

2 DATA UNDERSTANDING AND PREPARATION

2.1 Data Understanding

The dataset that we use for this project is the High Quality sensor data donated by Professor M. Kadous. The data is taken using 5 dimensional gloves that have position trackers which provide 6 degrees of freedom which are the x, y and z axis, as well as the roll, pitch and yaw information. The data signing has been done by Todd Wright, who signed 3 instances of each sign on 9 different days, for a total of 95 AUSLAN signs. The data was collected at a speed of 100 frames per second. The average number of frames for a sign in this dataset is 57. Some of the features in the data are roll, which depicts clockwise or anti-clockwise rotation of the hand; yaw, depicts the palm being straight ahead and pitch tells us whether the palm is pointing up or down. In addition to this, we have the bend measure for each of the five fingers for each hand, i.e. the thumb, forefinger, middle finger, ring finger and little finger.

2.1 Data Preprocessing

The raw high quality data that was received could not be used before preprocessing the data. We followed the steps to preprocess the data from the methodology used by H.Cate et. al. [2]. This process involves scaling of each sign in terms of number of frames, then spatially scaling the x, y and z axis sensor co-ordinates to be normalized and then, flattening the data for each sign to build the final matrix which is used for further analysis.

Temporal Scaling of the data is necessary because at each time, the instance of the sign has been performed under different conditions, as a result of which some signs are longer than the others. Performing this time-related scaling, all the sign instances now have the same length, which we have assigned as 57, which is the average number of frames for a sign. For this process, we resampled the data using Discrete Fast Fourier transformations, using the NumPy libraries in Python [4]. Figure1, depicts the original and the resampled versions of the x-axis signal of the signs Boy, Draw and What. As we can see, the resampled signals still retain the majority of information, which justifies our normalization process.

The sign's sensor data for the x, y and z co-ordinates has also been spatially scaled using the min-max normalization technique to take values between 0 and 1. This has been done to reduce the effect of different orientations while taking the same sign at different sign instances.

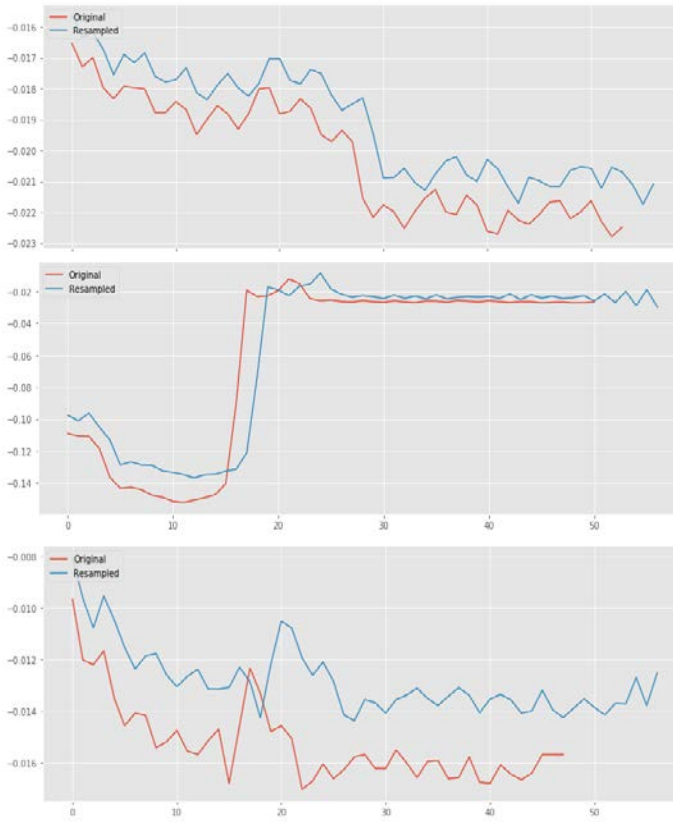

Figure 1: *Original versus Resampled signals – Temporal Scaling a. Boy b. Draw c. What*

The final step in the data preprocessing that we take is the time series flattening. Now since each of the sample for 95 signs has a normalized length of 57 frames per sign, we create flattened rows for each sign instance signed by a particular person which has NUM_FEATURES x NUM_FRAMES features. We also add the label for that sign, folder number of the person signing it, as well as the ID column as features to the dataset. The resultant data frame that we get is an N x D matrix, where N = NUM_SIGNS x SIGN_INSTANCES and D = NUM_FEATURES x NUM_FRAMES + 3.

2.3 Principle Component Analysis

Another experiment we performed with the dataset was to find out the pairwise similarity of signs and verify the similarity by looking at the actual signs from the AUSLAN sign bank. For this analysis, we computed the confusion matrix of the improved baseline model. Each element of this i x j matrix shows the number of elements of type i classified as j. We filtered out the diagonal elements which represent the same sign pairs. We then sorted the remaining elements in the decreasing order of highest misclassification number into another class. By doing this, we were able to map those classes which were frequently wrongly categorized as some other class. We found that the signs "crazy" and "think" were often misclassified as one another. Additionally, signs "different" and "hurry" were also interchangeably classified. This can be seen in Table 1 below:

	x	y	similarity
0	crazy	think	2
1	different	hurry	2

Table 1: Confused signs – Both Hands Data

In order to visualize this similarity, we performed a principal component analysis of the signs and created plots of their first principal components, as can be seen in Figure 2.

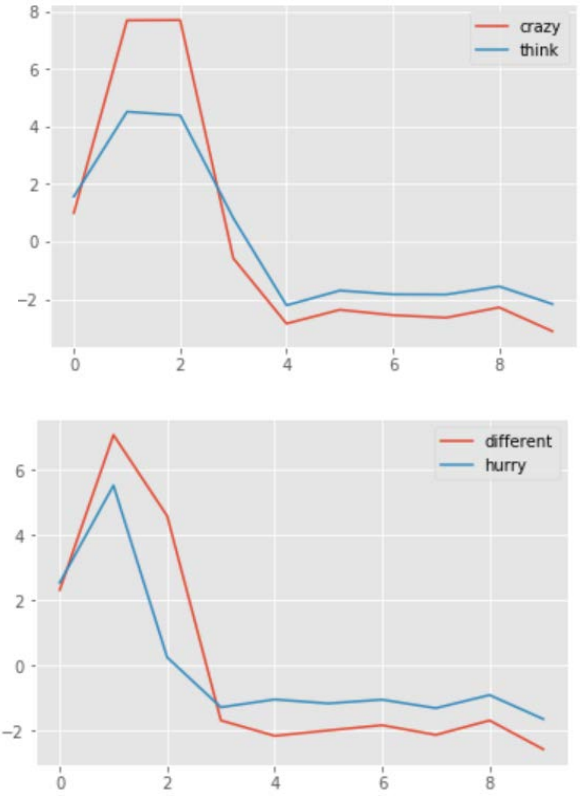

Figure 2: First Principal Component of each of the similar signs over 10 frames length

Furthermore, we performed the same analysis by taking just the right-hand data, and the similarity table that we computed is shown in Table 2:3

	x	y	similarity
0	give	thank	2
1	hurry	answer	2
2	later	write	2
3	responsible	love	2

Table 2: Confused signs – Right - Hand Data

From our experiment, we are able to conclude that the right hand itself is able to capture maximum variations and gives a steady performance, when compared to using data from both hands. The plots of the first three principle components of some of the above similar pairs are shown in Figure 3 on the next page.

Because our baseline model performs well on the data, this means that the 95 signs are linearly separable by an optimal hyperplane in the high dimensional space. Figure 4 depicts the 3- dimensional positioning of data points of 2 pairs of signs – 'Answer' and 'Norway', and the confused pair 'love' and 'responsible'.

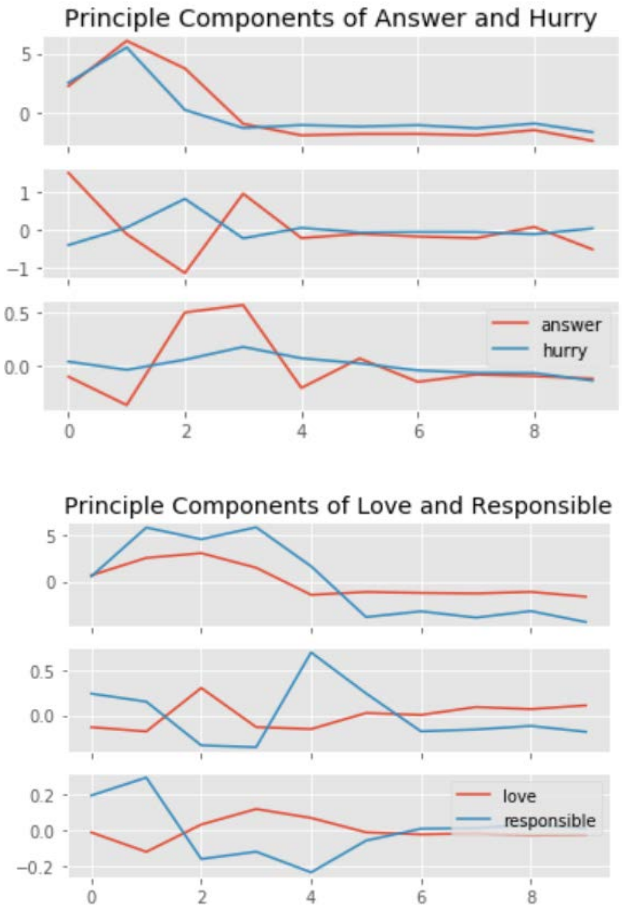

Figure 3: *Principle Component Analysis of Confused Pairs – Right Hand Data*

3 METHODOLOGY

3.1 Baseline Model

For the Baseline model, we started with implementing the SVM model by using the temporally scaled and flattened data. We use the SVC function from the scikit library in Python [4] which implements the "one-against-all" approach for the multi-class classification problem. For the test-train split, we have used 75% of the data to train the model and 25% of the data to test the model for each of the 95 sign classes. We implemented the model using both the linear as well as the RBF kernel. The RBF SVM model gives an accuracy of 91%, whereas the linear model gives a higher accuracy of 93%. We see that the linear kernel performs better than the RBF Gaussian kernel, even though the number of observations are greater than the number of features. One of the possible reasons could be that since the number of features is large, mapping to a higher dimensional space does not lead to improvement in performance.

Before building the linear SVM baseline model, we had the task of tuning the hyperparameter C. For this task, we did a cross validation for values of C in range [10-3, 10-2, 10-1, 1, 10, 102, 103]. The C value which has been used for the models is 10, as it had the highest cross validation score. For the RBF SVM model,

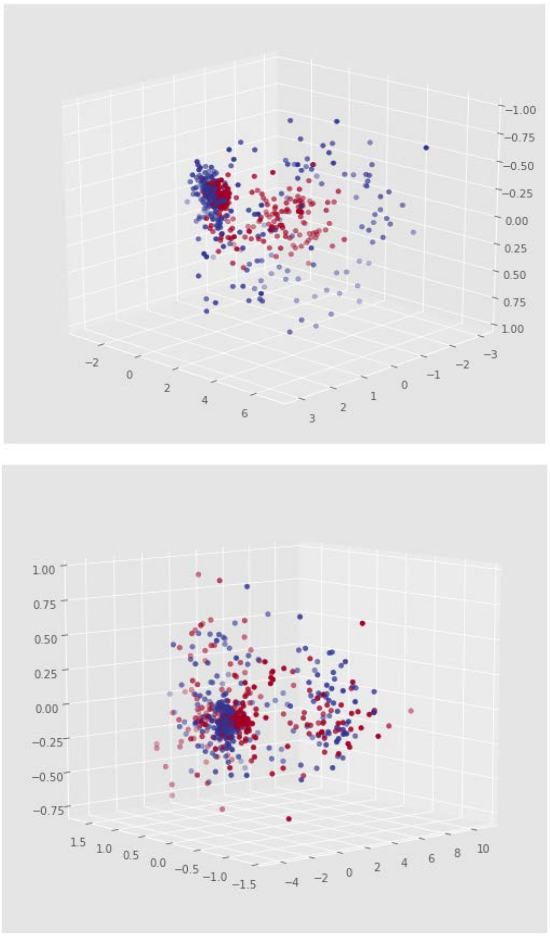

Figure 4: *Principle Component Analysis from Right Hand Data of i) Linearly separable pair(Answer, Norway) ii) Confused Pairs (Love, Responsible)*

hyperparameters C and gamma were tuned by Grid Search and cross validation to be 1000 and 0.001 respectively.

Furthermore, we were interested to find out the accuracy of our baseline model, which is the Linear Kernel SVM, when the data from only one of the hands was used. One of the interesting outcomes of this experiment was that the right hand's accuracy was 90.14%. On the other hand, using just the left-hand signals, the SVM model accuracy was just 35.41%.

This reinstates the fact that in sign language communication, individuals usually use only one of their hands consistently, which can also be called as the dominant hand.[5] Since left handers make up for only about 10% of the population [6], it helps us to corroborate why prediction accuracy is high with the use of right hand signals over the left hand. Since the brain's control over language and speech is concentrated to the left hemisphere for right handers, this revelation makes even more sense. Figure 5, on the next page, shows the confusion matrix plots for models built data from with left, right and both hands separately.

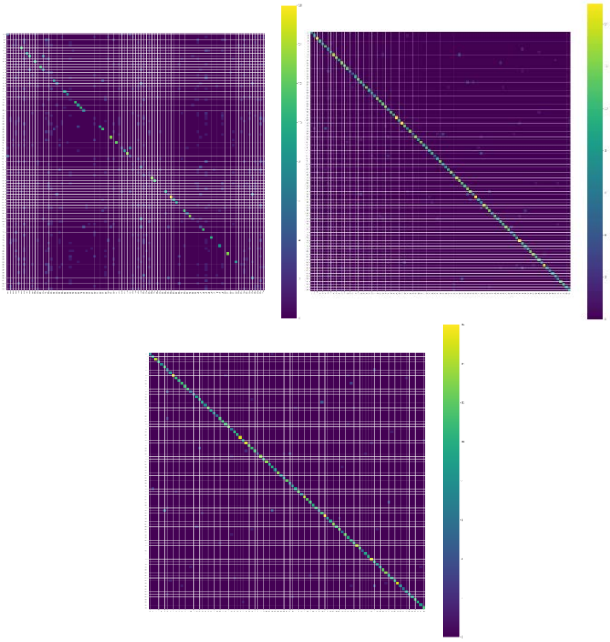

Figure 5: *Confusion Matrix for Baseline SVM Linear Model- Top (L to R): Signals from Left Hand; Signals from Right Hand, Bottom: Signals from Both Hands*

	C	Accuracy	Log (C)
0	0.001	0.089972	-3.0
1	0.010	0.545437	-2.0
2	0.100	0.892895	-1.0
3	1.000	0.936814	0.0
4	10.000	0.938375	1.0
5	100.000	0.938375	2.0
6	1000.000	0.938375	3.0

Table 3: Cross Validation to find out value of hyperparameter C

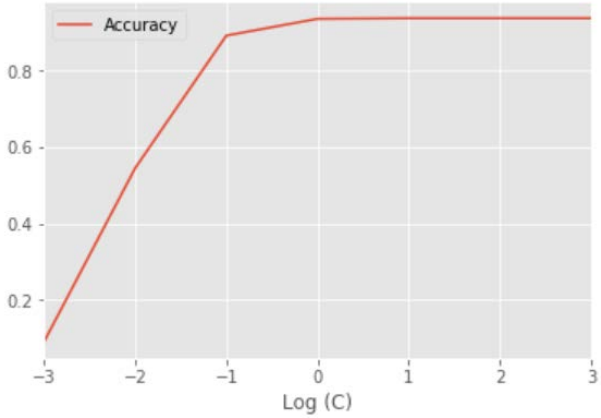

Figure 6: *Accuracy versus Log C to find out value of C by cross validation.*

3.2 Improvements in Baseline Model

As our initial approach, we tried to improve the baseline SVM performance by changing the spatial scaling method. Instead of the Min-Max scaling that we implemented earlier, which regulated the feature values to be between 0 and 1, we now scaled the raw signals using mean and standard deviation of each of the features in the raw signal.

We also did signal resampling using Fast Fourier transform (and then taking the inverse transform) and decreased the number of samples from the average length of 57 to the 10. We wanted to see if the same information is retained when we reduce the number of resampled frames. We conducted a 5-fold cross validation for the number of frames that must be resampled as the hyperparameter, and then calculated the cross-validation score by using a logistic regression model. The highest accuracy was when the number of frames to be resampled was 10. This can be seen in Table 4. Additionally, 25% of the data is used as the test set and 75% of the data is used for the training set.

	No of Frames	Cross Validation Accuracy
0	10	0.94067
1	20	0.93014
2	30	0.90211
3	40	0.88002
4	57	0.85584

Table 4: Cross Validation to find out value of hyperparameter 'No of Frames

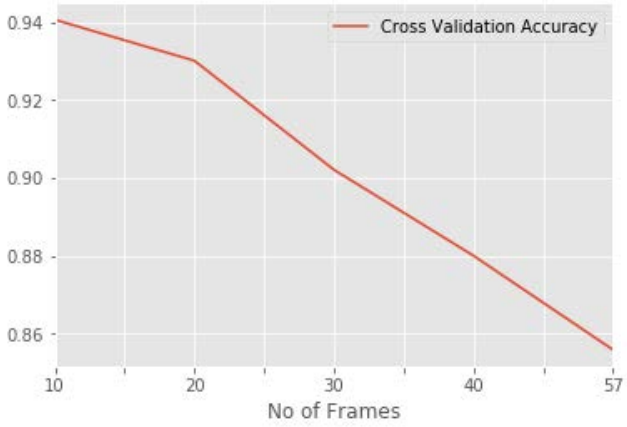

Figure 7*: Accuracy versus 'No. of Frames' to find out value of 'No. of Frames' by cross validation*

3.3 Logistic Regression Model

Since our baseline is performing better with a linear kernel, and through our PCA analysis, we have found that the classes are linearly separable in a higher dimensional space, we decided to implement a logistic regression model on the data. Since the feature space is large (NUM_FEATURES x NUM_FRAMES), we used L1 regularization to bring sparsity into the model. We used the 1:3 ratio of the test vs training set on the full data (both right and left hands) to get a cross validation accuracy of 96.55% and a test accuracy of 95.52%. The hyperparameter C (or 1/λ, where λ is the regularization multiplier) has been chosen as 1 after cross validation.

3.4 Random Forest Model with Meta Features

This model was implemented to study the effect of meta-features in temporal classification, a novel method devised by Kadous. [3] In his study, Kadous derives meta-features from the raw feature data and evaluate them for improving the existing algorithm.

During the analysis of features for running the random forest algorithm, certain signs were often being wrongly classified into other signs. For example, 'crazy' was the most misclassified sign, which was often misclassified with 'boy' and 'think'. A visual representation of the same can be seen in Figure 8. We can see that it only differs in the first few frames.
As a result of this, we implemented a Random Forest Model on the Right-Hand data, which uses only meta-features and none of the original features. We started with a minimal set of features including the mean, median, mode and Standard Deviation of the raw signal features. Then we made a grid of features which we thought could better define the raw signals. Further, meta-features were added iteratively using Grid Search and 5-fold cross validation. To implement this, we use the tsfresh [7] package in Python. Some of the meta-features we use are as follows:

1. Standard Deviation, Mean, Median and Mode of existing raw features.
2. Mean Autocorrelation, which is given by the mean correlation of the raw signal with a delayed copy of itself. [8]
3. Absolute sum of changes, which is given by the sum over the absolute value of each of the consecutive changes in the raw signal
4. Large Number of Peaks, which gives a Boolean output by checking if the number of peaks in the raw signal is greater than a predefined number, which is computed separately for each of the features in the signal.
5. Skewness, which is the skewness of the feature calculated using Fisher Pearson's standardized third moment coefficient.
6. Fingers and Thumb symmetry, which is motion symmetry of the thumb and fingers along the time series.
7. Mean second derivate central, which is given by taking the mean of the second derivative of the raw signals.
8. First and Last location of maximum, which is given by the relative last location of the maximum of raw signal.
9. Number of peaks

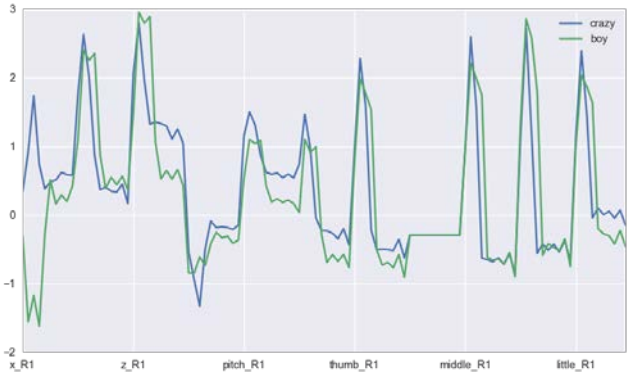

Figure 8*: Signal Crazy and Boy for right hand time series*

These meta-features of the right-hand data, when combined with the random forest algorithm give a prediction accuracy of 96.26% on the test data. The accuracy was tested using 5-fold cross validation on 75% of the data as training set and tested on 25% as the test set. Following parameters were tuned using grid search:

1. Number of estimators
2. Maximum depth of trees
3. Maximum number of features to consider at each node

3.5 K-Nearest Neighbors with Dynamic Time Warping

One-nearest-neighbor with Dynamic Time Warping (DTW) has been proposed as a difficult algorithm to beat when it comes to time series classification.[9] We implemented our version of k-NN classifier using DTW similarity as distance measure for this classification task. We compute DTW distance between each feature time series for a pair of right hand samples, and take L2 norm of computed feature distances. If A and B are two data samples, and j =1 to 11 represents feature indices for each of the 11 features of the right hand, DTW distance between the samples A and B is given by:

$$DTW(A,B) = \sqrt{\sum_{j=1}^{11} DTW(A_j, B_j)^2}$$

One drawback of using k-NN with DTW is that this approach is computationally demanding[9]. Another weakness of this classifier when working with multivariate time series is to assign appropriate weights to features. For this task, we have not considered assigning

importance to a particular feature. For example, uniformly weighted DTW distance defined above is not discriminative for signs 'happy' and 'make' as shown in Figure 8. This figure shows stacked time series for 11 variables for a pair of samples. Figure 9 shows frames from videos of these signs taken from Auslan Signbank[10]

From the videos, we find that the signs differ only in terms of finger bends, and thus weighting the DTW distances of finger bend measures might give us a more discriminative similarity measure. Due to high computational costs of k-NN, we run the classifier using the uniform weighted measure described above. Using cross validation, we find optimum value of k as 1, which is consistent with previous research.[9]. The 5-fold cross validation accuracy of 1-NN with DTW on 75% train set is 81.5%. Weighted k-NN makes for an interesting future scope of this model.

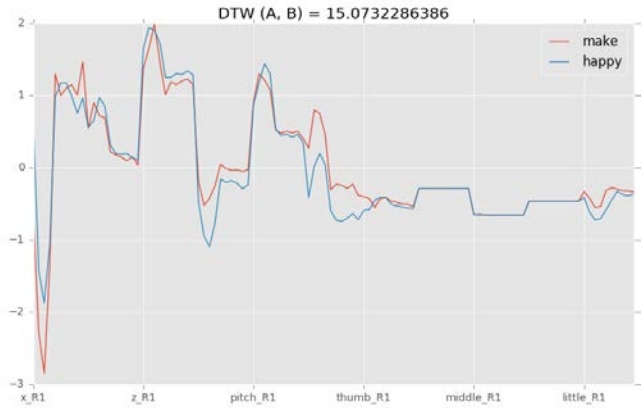

Figure 9*: Comparison of a pair of samples for signs "happy" and "make"*

Figure 10: *"Happy" and "Make" (L to R): Frame from video demonstration of "make"; Frame from video demonstration of "happy"*

3.6 Feed Forward Neural Network

We then start to build more complex models, which uses a simple feed forward network with hidden layers. In order to mimic the process followed by Logistic Regression The hidden layer that takes in the input, and then uses a 'SoftMax' activation function, which gives out the probability of each input, of being in one of the 95 classes. We have also added a Dropout layer to prevent model overfitting. The final layer is the output layer which gives the probability of the input signal to be in each of the 95 classes. The model takes in the flattened input raw signal, with all its features, resampled at 10 frames.

We performed parameter tuning of this model by tuning the number of epochs [20, 30, 50] and batch sizes [5,10,20] using Grid Search. The optimal batch size is 5 and the number of epochs is 50. This model gives a prediction accuracy of 92.94% on the test set.

3.7 Ensemble Learning – Weighted Soft Voting

We find that both linear classifiers and random forests work well on this dataset. The difference between the classifiers come from the feature space. The linear classifiers – SVM and Logistic Regression ignore the time series aspect and treat each frame as a feature. Random Forests, use meta features which explain the peaks and variance in time series. We decide to build an ensemble classifier that works on these two different feature spaces. We build 3 classifiers – SVM and Logistic Regression on the original time series feature space, flattened into frame features, and Random Forests on the meta features. We then take a majority vote of the 3 classifiers to make the predictions. Since we already calibrated our classifiers in previous steps, we use 'soft' voting technique, which predicts the class based on argmax of the sum of predicted probabilities. Our hypothesis is that some of the confused signs we saw in the discussion above, would be correctly classified by considering different feature spaces. We also tuned the weights to assign each classifier while voting using 5-fold cross validation. On 75% train dataset, 5-fold mean cross validation accuracy of the Voting Ensemble is 98.23% and test accuracy is 98.31%.

3.8 Future Scope of the Project

As a future scope of this project, we would like to work on low quality data and try to achieve better results on that data using the methods we have experimented with here. We would also want to explore 'feature-weighted' k-NN with DTW similarity and Hidden Markov Models that take into account the previous states for calculating likelihood. We also started working on a Recurrent Neural Network architecture, specifically Long Short Term Memory (LSTM) architecture. LSTM stores recent sequences in memory blocks and can handle sequence dependencies. Therefore, it is well suited for temporal classification tasks.

4 EVALUATION AND RESULTS

Our experimental results show that our Ensemble Learning model performs the best, with a test accuracy of about 98%. The model uses an ensemble of well calibrated classifiers trained on separate feature spaces, and uses soft voting to make predictions. Other models like the Feedforward Neural Network as well as the Baseline model perform well on the data. 1-NN with DTW does not perform well with similar signs. Feature weighting of DTW similarity might help discriminate better between these signs. The performance of different algorithms in terms of metrics like Precision, Recall, F1 and Accuracy values can be seen in Figure 11. An important conclusion of our analysis is that it is possible to achieve comparable performance with just using the right hand features. This can help save cost of sensor gloves for people who use sign language as well as reduce computation time for predictions. Downsampling number of frames to 10 retained the essential information to classify the signs, while effectively reducing the amount of training data.

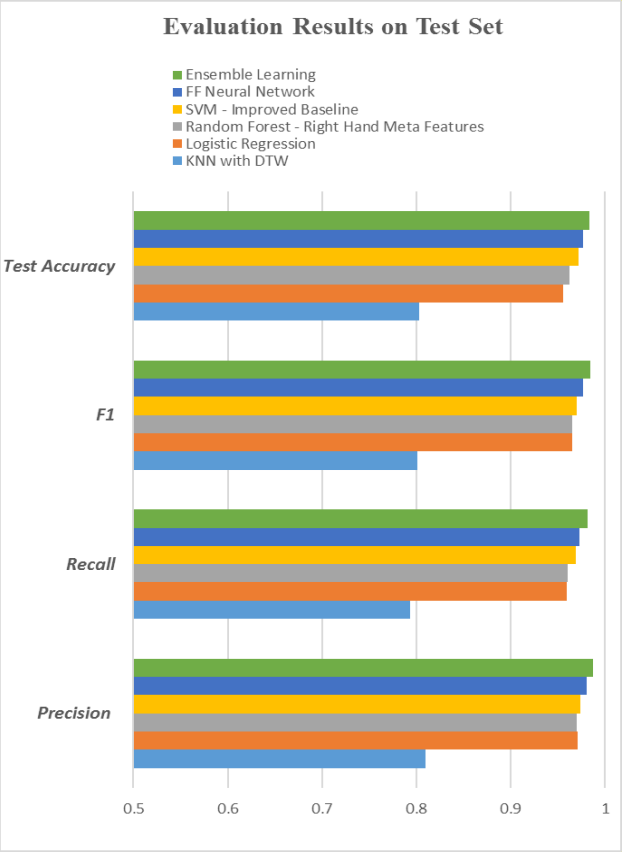

Figure 11: *Comparison of precision, recall, F1 score, and accuracy on test set between different models*

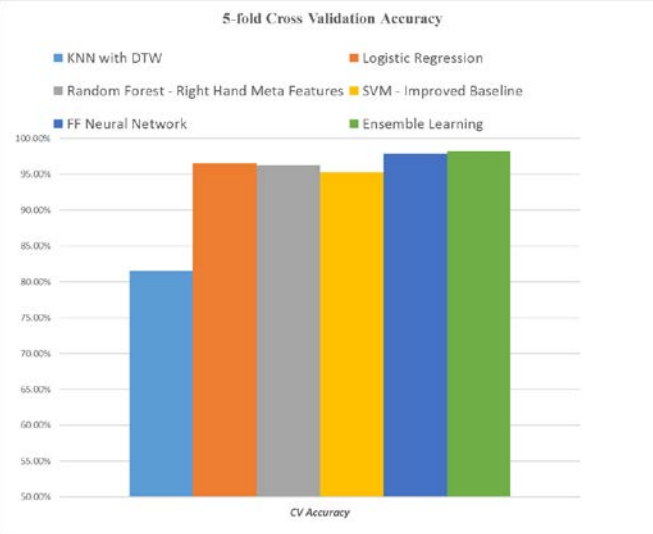

Figure 12: *Comparison of 5-fold cross validation accuracy on 7% training set between different models*

ACKNOWLEDGMENTS

We would like to thank Professor M. Kadous for donating the dataset and Todd Wright, the volunteer who signed for this dataset.

We would also like to thank Professor Leman Akoglu for her valuable suggestions throughout the course of this project.

A HEADINGS IN APPENDICES

A.1 Introduction

A.2 Data Understanding and Preparation

A.2.1 Data Understanding

A.2.2 Data Preprocessing

A.2.3 Principle Component Analysis

A.3 Methodology

A.3.1 Baseline Model

A.3.2 Improvements in Baseline

A.3.3 Logistic Regression Model

A.3.4 Random Forest Model with meta features

A.3.5 K-Nearest Neighbors with Dynamic Time Warping

A.3.6 Feed-forward Neural Network

A.3.7 Ensemble learning

A.3.8 Future Scope of this project

A.4 Evaluations and Results

A.5 Acknowledgments

A.6 References

REFERENCES

[1] Deafness and hearing loss. (n.d.). Retrieved April 10, 2017, from http://www.who.int/mediacentre/factsheets/fs300/en/

[2] Cate, H., Dalvi, F., & Hussain, Z. (2017). Sign Language Recognition using Temporal Classification. arXiv preprint arXiv:1701.01875.

[3] Kadous, M. W. (2002). Temporal classification: Extending the classification paradigm to multivariate time series (Doctoral dissertation, The University of New South Wales).

[4] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: Journal of Machine Learning Research 12 (2011), pp. 2825–2830.

[5] Vaid, J., Bellugi, U., & Poizner, H. (1989). Hand dominance for signing: Clues to brain lateralization of language. Neuropsychologia, 27(7), 949-960.

[6] Handedness. (2017, April 07). Retrieved April 10, 2017, from https://en.wikipedia.org/wiki/Handedness

[7] tsfresh.feature_extraction package (n.d.). Retrieved April 30, 2017, from http://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html

[8] Autocorrelation. (2017, April 25). Retrieved April 30,2017, from https://en.wikipedia.org/wiki/Autocorrelation#Estimation

[9] Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A., 2006. Fast time series classification using numerosity reduction. In: Proceedings of the 23rd International Conference on Machine Learning

[10] Auslan Sign Bank, Retrieved May 2, 2017 from http://www.auslan.org.au/