Classifying common hand gestures using accelerometer sensor data from a mobile device

**Ashley Hunt**   
Computer Science  
University of Nottingham

**Amit Kumar**  
Data Science  
University of Nottingham

**Shreeya Kumbhoje**  
Data Science  
University of Nottingham

*In this paper we attempt to utilise machine learning (ML) models and recurrent neural networks (RNN) to classify four common hand gestures (1. Drawing a circle, 2. Waving, 3. Gesturing come here, 4. Gesturing go away). These gestures were recorded using the accelerometer* *sensors inside a mobile phone. Our aim here is to train the computer using the labelled data so that it can classify between different hand gestures the user performs. The raw accelerometer* *data goes through a set of preprocessing techniques such as the application of a low-pass filter, Fourier transformation and then a feature extraction and selection process to optimise the data provided for the training of the model. We have deployed ML models including K-means* *Classifier, Support Vector Machines (SVM), Random Forest Classifier, XGBoost, K-Nearest Neighbour (KNN) and the Recurrent Neural Network Models Long Short-Term Memory layer (LSTM) and Gated Recurrent Unit (GRU) looking for an optimum fit model for multi class classification between gestures.*

*The outcome shows a good \_\_% accuracy for \_\_\_ ML model in comparison with other ML models implemented, and the accuracy of \_\_% LSTM(RNN model) and \_\_\_% for GRU(RNN model).*

# Introduction

Gesture recognition pertains to recognising meaningful expressions of motion by a human, involving the hands, arms, face, head, and/or body [1]. With a variety of application across different fields including monitoring activities and vital signs, enhancing gaming experience, and supporting applications in robotics and wearable technology. Our intent is to use smartphones which is a practically used device to establish communication possibilities between humans and machines differing from the traditional data collection methods. we further explore the usage of this technology to create accessibility features for individuals with disabilities which could be a potential project within the scope of man machine interaction.

In recent years, gesture recognition system has become very popular in the field of research, especially facial and hand gesture recognition system [2] gesture classification for hand movement can be classified into static and dynamic depending on the data acquisition method. In the past a significant amount of research has been carried out which investigates machine algorithms like Support Vector Machine [3] [2], Random Forest [3] , Hidden Markov Model which analyses data to discover patterns. A few Deep Learning (DL) models like CNN [3] [4] and some hybrid models like CNN-LSTM [3] mostly focusing on image processing, which has now made it a common subject in the domain. However, a limited amount of research focuses on using signal data collected as a dataset using a normally accessible device among humans today.

# Background

Over the years a significant amount of research has been carried out for activity recognition and for different purposes. Each study differs in the factors it considers during data collection, the types of activities performed, the nature of the dataset (whether static or dynamic [2] [4]), and the algorithms used for classifying gestures. As data collection plays an important role, it necessitates the requirement for selecting the right way to data acquisition. Modern smartphones and smartwatches are equipped with sensors. Gyroscope, accelerometer, magnetometer, temperature, and sound sensors have been used for activity detection [5]. The widespread use of mobile phones globally, along with their extensive capabilities, was considered for their potential to act as remote controllers, enabling disabled individuals to perform various tasks. The data was collected using the Phyphox application compatible on both android and IOS devices and the study integrated accelerometer data from 4 different human activities [6].

The static time series 3D dataset was then explored for multi class classification. The most adopted classifiers included the k-nearest neighbor (KNN), random forest (RF), multilayer perceptron (MLP), support vector machine (SVM), decision tree, and artificial neural network (ANN),CNN, etc. [5]. SVM usually needs careful feature extraction and selection, as well as other traditional ML algorithms like naïve Bayes (NB), K-nearest neighbors (KNN), and decision tree (DT) . HMM is memoryless and unable to use contextual information. [7]. However, DL algorithms worked best on raw normalised data. Next, the algorithm selection which depends on the variety of data to be processed . For image based data algorithms like CNN, CNN-LSTM are suggested. SVM [3] [2] [1], RF [3] were discovered as the most popular ones for gesture classification, the best results in precision-recall-FI (abbr.: P-R-F1), followed by the KNN and LR [8]. The popular models among the DL were RNN’s GRU.

# Methodology

To achieve our aims of producing a classification model for four gestures (come, go, wave and circle), the raw accelerometer data is heavily pre-processed and undergoes several key transformations to contribute to the classification process. Only the most useful features are taken through to the model training phase.

## Data Collection

Data was collected using the Phyphox mobile application[[1]](#footnote-1). Each member of the team performed the same four gestures for eight iterations per recording, and five recordings each per gesture – 40 iterations per person per gesture for a total of 480 complete gestures.

These gestures were initially recorded continuously, without stopping, as per the coursework specification but after trial and error we discovered that segmenting the data was a lot more difficult when there were no clear spaces between the gestures. Instead, we opted to leave a distinct one second gap between performing each gesture in each recording. Adding these pauses allowed us to segment the recording more easily into its separate gesture patterns.

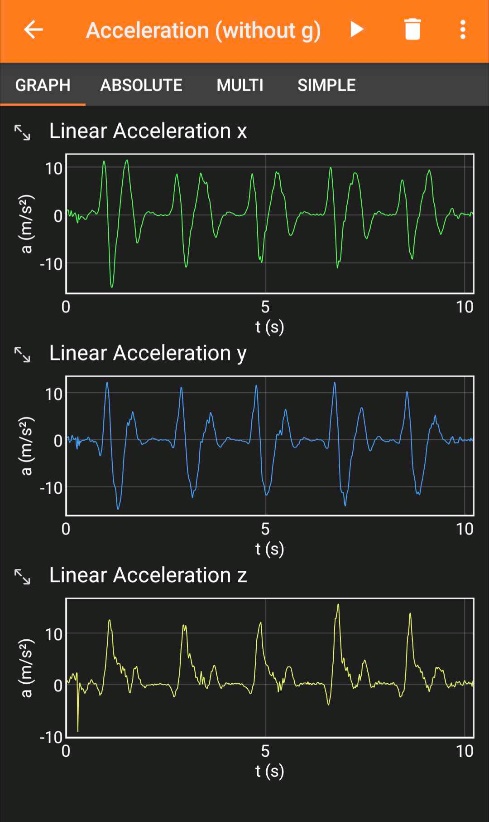


Figure 1: Screenshot of the Phyphox recording interface.

We imported two separate data sets, one for testing and training and the other for validation. Both data sets would undergo the same pre-processing, but we would apply a *train\_test* split to the first and the second would be kept away from the model until we needed to perform validation on the model performance.

## Data Exploration

Once the data was recorded and the associated csv files were imported into *pandas* *DataFrame* we applied a basic trim to the data. We would inspect, from the beginning of the file, every 20 data points and if the mean acceleration over that sample was < 0.3 then the data was removed. We did this from the beginning of the file as well as the end of the file. This would help us reduce unnecessary noise at the beginning and end of the file, making segmentation easier.

We then began to explore the data by inspecting the raw sensor data and creating visualisations of the data. These visualisations would allow us to understand the data and what features of the data would play a significant role in classification.

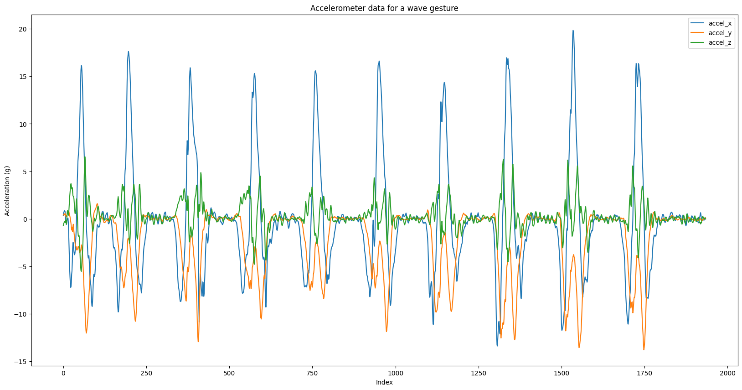


Figure 2: Raw accelerometer data for a wave gesture recording.

## Data Pre-Processing

To balance the dataset, ensuring that the model would not prioritise one gesture, we found which gesture had the least number of recordings and then dropped all recordings greater than that number. This step would ensure that each gesture contributed equally to the model training.

To begin our pre-processing, we used the *scipy.signal* *butter* and *filtfilt* functions, applying a low-pass frequency filter to the data. This filter would apply dampening to the sensor data and reduce noise. Visually, this made the waveforms smoother and less jittery. Filtering ensures that the model does not overfit based on the noise in the system but uses the more generic shape of the data.

We also applied an exponentially weighted moving average (EWMA) filter at this stage. EWMA filtering was performed so that spikes in the data were dampened. This dampening was especially important when considering recordings that contained many different types of gestures as some gestures tended to have much higher acceleration than other, simply due to the nature of the gesture.

Next, we applied a Fourier transformation to the data using *numpy Discrete Fourier transform (FFT)*. Fourier transformation can be used to decompose a signal into its underlying frequencies, which can be useful for removing noise from the accelerometer data as well as determining the most important parts of the signal that define the gestures.

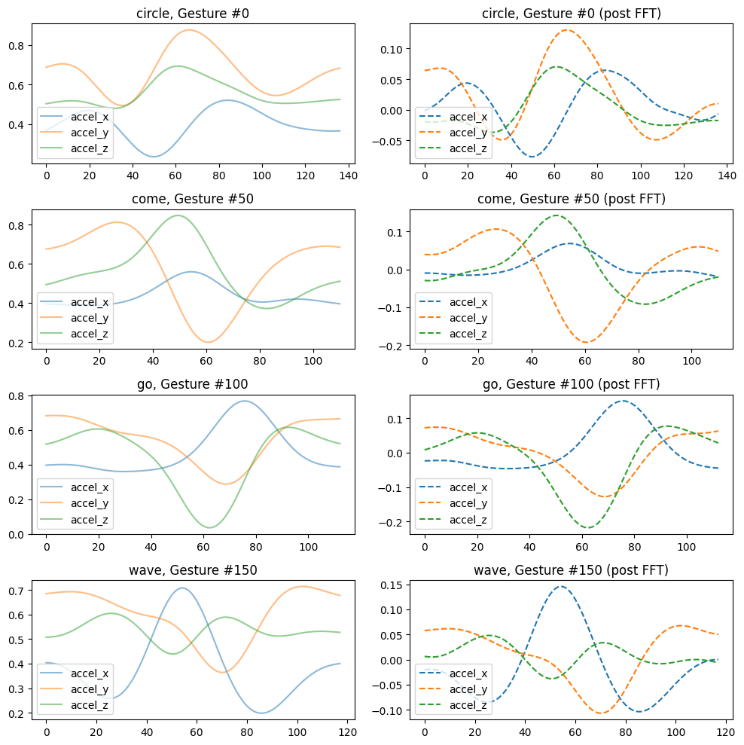


Figure 3: Example gesture data before and after Fourier transformation and filtering.

Applying this Fourier transformation filtering in this way meant that the signals were more uniform and less prone to noise. It was expected that a simpler wave signal would translate into a more accurate and reliable model.

Dynamic time warping (DTW) can also be used to find the similarities/distance between different wave signals, which can be used to shift rotate the waves so that slight misalignments in the time domain are not factors in the segmentation and eventual feature extraction of the gestures. DTW was considered a stretch goal for the project but was not implemented.

## Segmenting the Data

The next step of processing was to separate the individual gestures from the files that contained six to nine gestures of the same type. This would be required so that each gesture was separated in its own block of data to be associated with a label. Our first attempt to apply this separation was to visually inspect the data and its rolling average and deviation – this would help us identify an appropriate value to decide on when one gesture was starting, and another was ending.

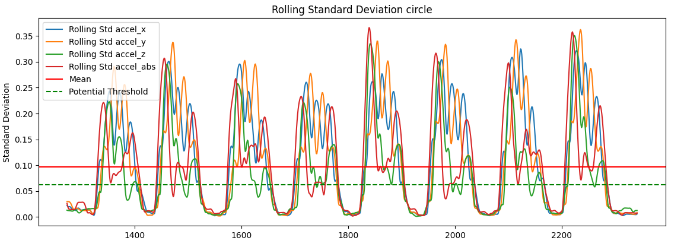


Figure 4:Rolling standard deviation of a circle gesture recording.

This figure outlines our suggested splitting point (any points where the acceleration standard deviation is lower than the threshold, in green). Using these points, we could segment the file into its separate gestures ready for feature extraction.

An alternative approach was then adopted where we utilised *scipy.signal find\_peaks* to automatically detect peaks in absolute acceleration, and from those peaks expand its boundaries until the acceleration was below some threshold (indicating the end of the gesture).

A graph of a graph

Description automatically generated

Figure 5: Absolute acceleration graph for a circle recording, showing the peaks in the data that represent the gestures.

We could then visually inspect the gestures to ensure that the segmentation was being correctly applied.

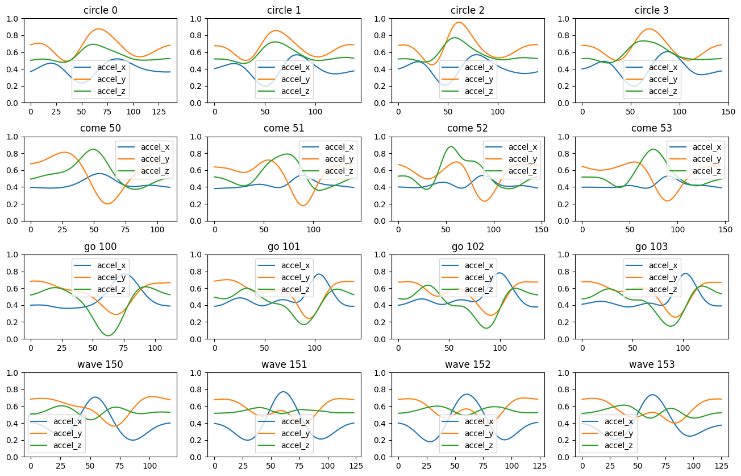


Figure 6: A preview of the segmented gesture data.

Once the gestures were separated, we once again performed dataset balancing to ensure that each gesture had an equal number of observations and did therefore not contribute more to the model training.

## Normalisation

We then applied a *scikit-learn* *MinMaxScaler* to the data – which transforms the data to values between 0 and 1 whilst maintaining the relationships between the datapoints, this is done using the below formula:

We performed this transformation to ensure that the machine learning models did not perceive higher values as more important for training.

## Feature Extraction

For the model to accept our data we would need to transform the data once again so that each gesture’s data was the same shape. To achieve this requirement, we would segment the data, this time with each gesture being broken up into distinct sections with some overlap. Then using these slices of data, we could extract key features such as: mean, standard deviation, median, skew and kurtosis. This would result in data that is a summary of each slice of the gesture. These features could then be visualised and taken through the feature selection process.

## Feature selection

Once all features were extracted, we visually inspected the averages of the features to find the features that had the clearest disparity between the gestures. Feature selection is critical to reduce the computational and time cost of classification, as well as increase the model’s accuracy.

A graph of different colored lines

Description automatically generated

Figure 7: Feature extraction/visualisation for the mean of each segment.

A graph with different colored lines and dots

Description automatically generated

Figure 8: Feature extraction/visualisation for the standard deviation of each segment.

Originally, we had planned to use these visualisations to find features that provided clear separation between the gestures and to drop the features that did not provide strong correlation. However, this process allowed room for ambiguity and did not allow easy automation. Instead, principal component analysis was used to automate the process of feature aggregation and selection.

A group of graphs with different colored dots

Description automatically generated

Figure 9: Principal component analysis visualisation for n=3.

After experimenting with various values of *n*,the accuracy for model training was best when the dataset was reduced to just two components.

After applying principal component analysis k-means clustering was utilised to ascertain how well the PCA reduced feature set could be clustered. This visualisation would allow us to see how well the clusters were separating the data based on different values for *n*.

A diagram of a diagram

Description automatically generated with medium confidence

Figure 10: K-means clustering visualisation for four gestures.

## Train/Test/Validation Data Splitting

Before we began training the models we split our data using *sklearn train\_test\_split.* This function would split our data into data for training our model and then keep separate data for testing our model. Ensuring that the test data remains completely unseen by the model during training is critical. Data leakage can occur if any step prior to model training involves knowledge about the test set. This can lead to overfitting and falsely optimistic performance metrics.

## Model selection

Based on our initial research of similar papers we had created a shortlist of potential models to evaluate for our requirements. Firstly, we considered machine learning models and then neural networks.

### K-nearest neighbours (KNN)

KNN is very simple to understand and implement. The algorithm classifies new data points based on the majority vote of the k nearest points in the feature space. This makes it a straightforward choice for initial experimentation and prototyping. KNN is also an effective classifier when dealing with small datasets such as ours, due to limited number of gestures we could record.

### Random Forest

A *RandomForestClassifier* was used with a predefined random state for reproducibility. Grid search was applied to fine-tune hyperparameters such as the number of estimators, maximum depth, minimum samples split, and minimum samples leaf. Cross-validation (cv=3) was used with balanced accuracy as the scoring metric.

### Extreme Gradient Boosting (XGBoost)

In our exploration of suitable models for gesture recognition using accelerometer data, XGBoost emerges as a strong candidate. XGBoost, or Extreme Gradient Boosting, is an implementation of gradient boosted decision trees designed for speed and performance. It is particularly well-suited for handling non-linear relationships and complex patterns in data, characteristics often present in gesture recognition tasks. Furthermore, XGBoost includes built-in regularisation, which helps prevent overfitting, a common challenge in machine learning scenarios involving complex models and limited data.

### Long Short-Term Memory layer (LSTM)

LSTM is a recurrent neural network which is used for sequence prediction, such as involving time-series data. LSTM can capture temporal dependencies which make them effective for recognition gesture recognition tasks. In our implementation, we preprocess the data, encoded the gesture labels using *LabelEncoder* to convert them into numerical labels, data are then standardise using *StandardScaler*. Subsequently, the number of time steps of windows representing the quantity of previous time steps of data that will be utilised to predict the current gesture is defined. The data is then prepared for model training using a custom function which generates sequences of data samples based on the time steps defined earlier.

LSTM sequential model is built using *Keras* which consisting of two LSTM layers with 50 neural units each followed by a Dense layer having softmax activation for multi-class classification. Next the model is compiled using the Adam optimiser and sparse categorical crossentropy loss function. The model is then trained and validated using the training and validation set.

### Gated Recurrent Unit (GRU)

Unlike traditional machine learning models that treat data points as independent, LSTM and GRU can learn from the progression and duration of gestures as represented by the time-series data from accelerometer sensors. This capability allows them to minimise the impact of temporal gaps and variances in gesture speed. Furthermore, GRU offers a more streamlined architecture that requires fewer parameters than LSTM, potentially increasing computational efficiency without significantly sacrificing model performance. A simpler model with fewer parameters is also a good property since we only have limited gesture data. By leveraging these models, our approach can more accurately and robustly classify the gestures.

# Results

Several machine learning models achieved a high, near 90% validation accuracy whilst the deep learning models struggled with the limited amounts of data that was available.

## Machine learning Models

### K-nearest neighbours (KNN)

Utilising the best hyperparameters found:{'knn\_\_metric': 'euclidean', 'knn\_\_n\_neighbors': 3, 'knn\_\_weights': 'uniform'} an accuracy of 0.98 was achieved on testing data and 0.87 on validation data.

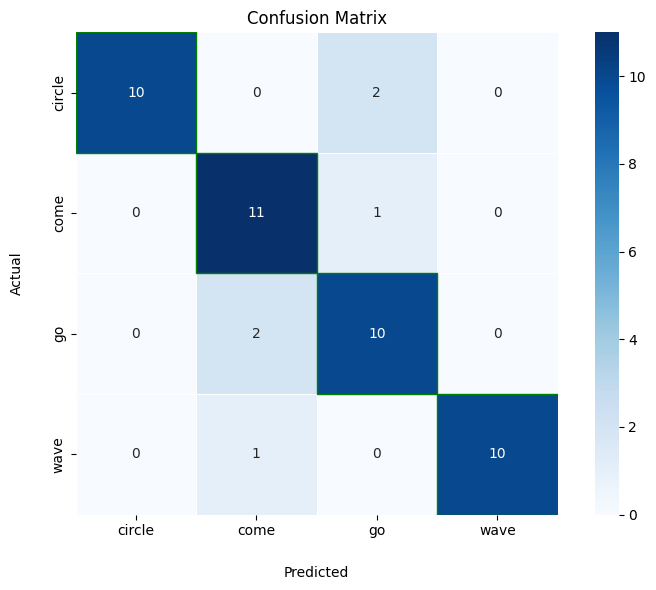


Figure 11: KNN validation results confusion matrix.

### Random Forest

The best parameters obtained from the grid search were {'n\_estimators': 250, 'max\_depth': None, 'min\_samples\_split': 10, 'min\_samples\_leaf': 2}. The optimal model was then used to predict gestures on the testing set. The test set achieved a balanced accuracy of 0.98, thereby providing an indication of the model's overall predictive ability. For validation this accuracy was reduced to 0.85.

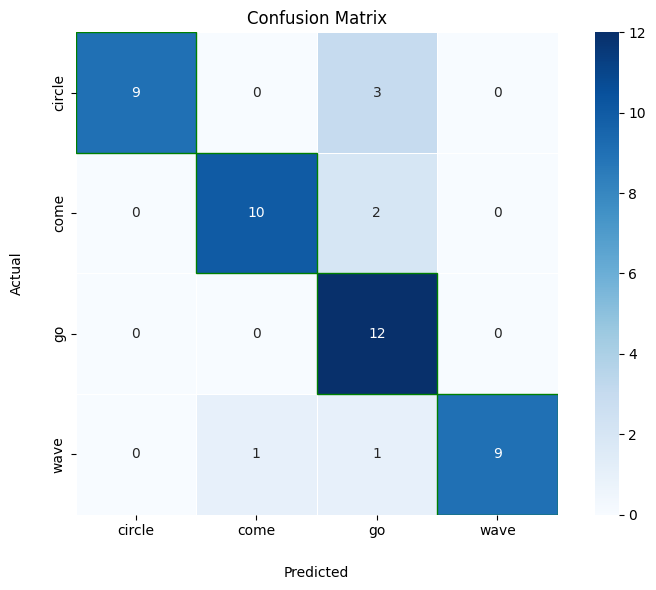


Figure 12: Random Forest validation results confusion matrix.

### Support Vector Machines (SVM)

The data was pre-processed using standard scaling, and a grid search with cross-validation was carried out to optimise the SVM's hyperparameters, focusing on the regularisation parameter C, kernel type, and gamma value. The optimal parameters were C=0.1, using an 'rbf' kernel, and 'scale' for gamma.

After training, the SVM model achieved a balanced accuracy of approximately 0.98 on the test set, indicating its ability to generalise to unseen data. On validation data this accuracy was reduced to 0.89.

It's worth noting that the process involved substantial computational resources, and attempts were made to fine-tune the parameters by experimenting with various kernel types and C values, but these adjustments did not yield significant improvements in performance.

Overall, accuracy for SVM model has shown its potential for gesture classification, although further exploration, feature engineering, or alternative algorithms could be considered to enhance accuracy further.

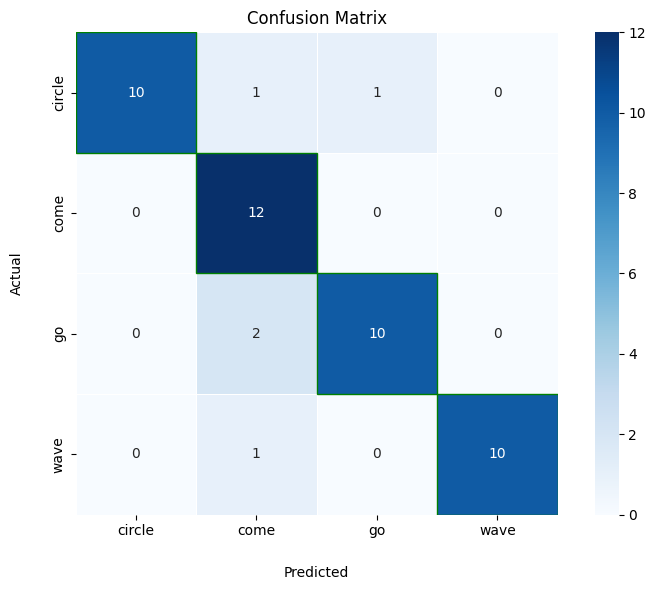


Figure 13: SVM validation results confusion matrix.

### XGBoost

The XGBoost classifier was used for gesture classification on the dataset. The dataset was divided into training and testing sets, and a grid search was performed to optimise the hyperparameters of the XGBoost model. The parameters used were number of estimators (trees) in the ensemble, maximum depth of each tree, learning rate, and subsampling ratio. The best parameters identified through the grid search were an ensemble of 300 trees with a maximum depth of 20, a learning rate of 0.1, a subsampling ratio of 0.8.

After training, it achieved a balanced accuracy of approximately 0.98 on the test set. This performance indicates the model's ability to classify gestures based on the features. On the validation data this accuracy decreased to 0.85.

In comparison to SVM, the XGBoost model demonstrated a higher balanced accuracy, suggesting its potential as a robust classifier for gesture recognition. Further exploration could involve feature engineering, or additional parameter tuning to potentially enhance classification performance even further.

## Neural Network Architectures

### Gated Recurrent Unit (GRU)

Write up goes here…

### Long Short-Term Memory layer (LSTM)

Write up goes here…

# Discussions

The model with the highest accuracy was SVM with 0.89 validation accuracy, closely followed by KNN and RF.

As outlined in the results section, machine learning models outperformed the deep learning models, but this was likely due to the limited number of data samples that we had collected. It is expected that with more gesture data we could achieve accuracies over 90% with deep learning models such as LSTM.

To gain more accuracy with our models we think that we would need to improve our gesture segmentation process somehow to ensure that the entire gesture is segmented correctly, even when there is distortion in the accelerometer data or excessive noise.

Higher accuracies could also be achieved by aligning the gesture signals using dynamic time warping of the signals which would improve the feature extraction and the overall representation of the gestures.

# Conclusion

Classifying gestures from mobile device sensor can be difficult due to the variation in movement that is possible from user on a mobile device and the number of ways in which a gesture can be expressed, both by the individual and by the digital representation of it.

# References

|  |  |
| --- | --- |
| [1] | S. &. A. T. Mitra, “Gesture Recognition: A Survey,” IEEE, 2007. |
| [2] | M. &. A. J. HafizurRahman, “ Hand Gesture Recognition using Multiclass Support Vector Machine,” *International Journal of Computer Applications,* p. 6, 2013. |
| [3] | S. S. M. K. F. F. B. E. G. C. Sara Moccia, “Automated classification of hand gestures using a wristband and machine learning for possible application in pill intake monitoring,” *Computer Methods and Programs in Biomedicine,* p. 10, 2022. |
| [4] | M. S. H. R. u. I. a. K. A. M. Z. Islam, “Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation,” IEEE, USA , 2019. |
| [5] | U. F. I. A. K. A. U. R. P. S. S. H. K. K. Siddiqui UA, “Wearable-Sensors-Based Platform for Gesture Recognition of Autism Spectrum Disorder Children Using Machine Learning Algorithms,” 2021 May . |
| [6] | A. L. W. P. C. C. R. S. V. H. Odhiambo CO, “Detecting Medication-Taking Gestures Using Machine Learning and Accelerometer Data Collected via Smartwatch Technology: Instrument Validation Study,” JMIR HUMAN FACTORS , . , 2023 May . |
| [7] | C. H. L. W. L. Y. L. C. Zhao S, “ Hand Gesture Recognition on a Resource-Limited Interactive Wristband,” 2021. |
| [8] | I. &. M. J. &. G. T. &. V. M. &. B. M. Stancic, “Comparison and Evaluation of Machine Learning-Based Classification of Hand Gestures Captured by Inertial Sensor,” 2022. |
| [9] | D. &. F. H. &. R. R. &. M. C. Anggi Maharani, “Hand gesture recognition using K-means clustering and Support Vector Machine,” 2018. |
| [10] | Y. N. a. K. Wohn, “Recognition of space-time hand-gestures using hidden Markov model,” New York , 1996. |
| [11] | D. K. G. a. S. Ar, “Static Hand Gesture Recognition Using Mixture of Features and SVM Classifier,” 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior , India , 2015. |
| [12] | T. &. K. S. &. T. N. &. M. V. J.Joshi, “Static Hand Gesture Recognition using an Android Device,” International Journal of Computer Applications, 2015. |
| [13] | A. &. J.-T. J. &. T. J. &. P. A. &. L.-G. A. &. C. R. A. Toro-Ossaba, “ LSTM Recurrent Neural Network for Hand Gesture Recognition Using EMG Signals,” 2022. |
| [14] | Z. Y. a. X. Zheng, “Hand Gesture Recognition Based on Trajectories Features and Computation-Efficient Reused LSTM Network,” *IEEE Sensors Journal ,* vol. 21, 2021. |
| [15] | I. &. M. J. &. G. T. &. V. M. &. B. M. Stancic, “Comparison and Evaluation of Machine Learning-Based Classification of Hand Gestures Captured by Inertial Sensors,” 2022. |

1. https://phyphox.org/ [↑](#footnote-ref-1)