Arm Exercise Guide for Post-Stroke Patients

AML Lab Project Final Report

Chatchanan Varojpipath, Irfan Kustandy, Jingchao Zeng

31 March 2021

Abstract

Assessments of post-stroke rehabilitation exercises by clinical expert is not always available or accessible to the patients due to some limitations or constraints. A presence of an intelligent system using machine learning (ML) approach that can guide through the exercises by classifying them and providing useful feedback can be a great help to tackle the issue, without the need of full supervision from a therapist. This is especially targeted for mild-symptom patients who can self direct their movements. Hence, keep tracking the progress towards recovery can be achieved.

To approach this problem, a wearable inertial measurement unit (IMU) platform is designed. 6-axis IMU sensor readings and Euler angle of 4 exercises, such as Straight Push, Rotation, Bicep Lift and Arm Swing, are visualised and analysed. The failure cases, including low-speed, shaking and partial movement are also considered to provide useful feedback. After preprocessing data, 1600 datasets are collected and annotated manually for model training. The multilayer perceptron (MLP), k-nearest neighbors algorithm (KNN), Decision Tree (DT) and Support Vector Machine (SVM) model are trained and validated. Bluetooth Low Energy (BLE) communication is applied to remove the wired connection and offer the user flexibility to do exercises. A graphical user interface (GUI) platform is developed for the user to interact and provide the performance analysis in each exercise.

The linear SVM is chosen as the optimal model and later implemented into the final device. Finally, experimental testing is carried out to validate the reliability of the device where many subjects are invited to perform all exercises. The final classification accuracy for members and one external subject is 94% and 84%, respectively.

Contents

1 Introduction

1.1 Background and Context

The applications of ML approach in healthcare sector have been substantially advancing over many years. Building an intelligent system with ML techniques for improving conditions of the patients has always been appealing implementations that can bring beneficial measures in real life scenarios. The system can also be more accessible to the patients without having to get a full supervision from the therapists.

The motivation of our project is inspired by post-stroke rehabilitation for upper-limb body part, specifically the arm. The targets are patients with mild-symptoms who can self-direct their movement. The idea is to implement ML for arm exercise classification into a simple embedded system using Arduino. Moreover, we would like to provide useful feedback and suggestions to the users based on the classification result, for their improvements towards recovery.

1.2 Scope and Objectives

The main objective of this project is to implement ML techniques into a Arduino 33 Nano BLE sense for arm exercise classification wirelessly with good accuracy, and to evaluate the results in real-life scenarios, in order to provide feedback to users as a guide for post-stroke rehabilitation. The scope includes the following:

- Assembly suitable hardware components that fully functioning
- Collect datasets with minimum bias
- Prepocess datasets and extract features to feed the model
- Select, train, validate, and test the ML models
- Implement and test a ML model to Arduino
- Apply BLE communication and design GUI interface as a user application
- Invite friends or patients to verify the robustness of the whole system

1.3 Achievements

The important milestones that have been achieved in this project are listed as follows:

- **fully functional hardware** assembly from each team member.
- dataset collection of 16 exercises with a total of **1600 recorded movements** from all team members.
- successful **training and validation** of some ML models such as MLP, KNN, DT, and SVM.
- a working ML model of Support Vector Machine (SVM) with linear kernel that achieves **94% accuracy** of performance result, chosen as the best model.
- **GUI** web with BLE communication for users' application.

1.4 Overview of Dissertation

Following this Introduction, this report contains a literature review section which collaborates on the stroke rehabilitation and ML topics from reference sources. The Hardware section shows the device components for this project. Data Collection section explains the methods for gathering dataset such as filtering and data analysis before feeding to ML models. In Methods section, we present 4 main ML models we choose for exercise classifications, then implement them on Arduino for real-life tests, along with the development of the GUI for the whole system using BLE communication. The whole system testing results are presented in the Experimental Results section. The Workplan sections shows the overall timeline of this project for both planning and realisation. Finally, the Conclusion section summarises the whole aspects of this project, including evaluation and future work for improvements.

2 Literature Review

2.1 Stroke Rehabilitation and Assessment

Stroke is the main cause of dysfunction in the United State, hurting nearly 1 million people annually. Majority of stroke patients have significant mobility impairment in their upper extremities, which negatively affects their activities in daily life such as bathing, eating, and dressing [1]. Other common symptoms are facial weakness, arm weakness, speech disturbance, and headache [2]. Thus, physical therapy intervention is crucial for stroke survivor to reduce motor disability. The purpose of stroke rehabilitation is to help patients regain lost abilities when stroke damages parts of patient's brains. The independence and improvement of patients' quality of life are expected after recovery program where stroke survivors participate in well-defined physical activities. However, the severity of stroke damage varies greatly and depends on the part of body affected by stroke. Therefore, a particular rehabilitation program is crucial and required for patient with certain neurological disorders. To address this variation, task-specific exercises are necessary and considered to be effective ways for post-stroke patient to prevent neurological disability and enhance their mobility function [3].

In addition, an assessment of rehabilitation is equally important to yield meaningful evaluation of a patient. Typically, the process of assessment for stroke recovery is carried out by experienced therapists in a facility for the rehabilitative therapy [4]. Task-oriented activities are normally prescribed for post-stroke survivors as in-home rehabilitation. However, in absence of such therapists, patients at home normally suffer from a lack of appropriate evaluation and are not able to make valid judgement. Furthermore, in-home patients may become unmotivated or confused to follow specified prescription without the feedback of a therapist [5].

This motivates us to develop an end-to-end system that offers patients an opportunity to perform rehabilitative exercises and subsequently provide users with constructive feedback which is used for self-evaluation and help reduce the need for staying in inpatient rehabilitation facilities. Moreover, several existing methods uses computer-assisted support tools and motion capture technologies to assess the quality of patient's exercise. There are two issues with this approach. The first one is that users are required to stay in

front of motion sensor in order to be successfully captured in the frame, which introduces extra burden to a user where physical exercises need to be performed in specific location, and sensors need to be attached to various body joint location. The second drawback is the accessibility and affordability of such tools, which may not be available to lower class citizens. Inspired by these problems, we propose affordable Arduino-based device that does not pose any restrictions on performing location and is fully available to every patient. The proposed device is also compact and portable, which greatly motivates patients to participate in rehabilitation exercises.

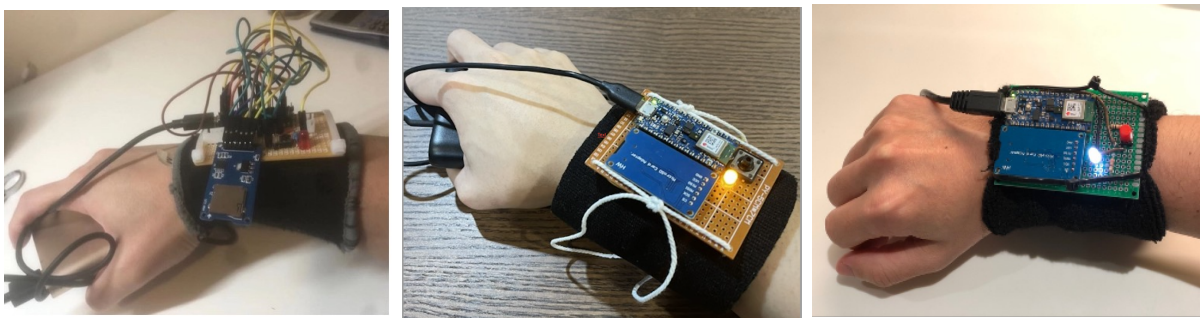
2.2 Machine learning model

In order to boost the performance of any practical application, especially in healthcare domain where highly accurate results are crucial. Machine learning-based techniques are widely adopted and incorporated into a system. [6] used SVM to classify level of severity in symptoms for patients who suffer from neurological disorders (e.g. Parkinson), and [5] exploited LSTM networks to give quantitative results based on the quality of stroke rehabilitation exercises. Nowadays, most existing methods use deep learning approach which is known to be powerful in automatically extracting features and scale well to unstructured tasks. However, in healthcare domain, deep learning method often achieves accuracy below 90% [1, 7], which is considered to be satisfactory results for general classification tasks. However, we realize the importance of higher accuracy in post-stroke rehabilitation exercises where few misclassified can be very costly for post-stroke patients.

Therefore, we adopt traditional ML techniques (SVM), couple with handcrafted statistical features. This technique is possible in our cases where each exercise possesses relatively unique characteristics, and well-structured procedures for each exercise are established. Despite the popularity of deep learning-based system for healthcare problems [7, 8, 9], SVM has seen numerous use case in quantitative measurement of motor symptoms due to its high performance. In particular, [6] used SVM to assess movement disorder in patients suffered from Parkinson's disease with an average accuracy of approximately 90%. Non-linear SVM was also used in [10] as a classifier for assessment of stroke patients with finger movement problem with 95% confdence interval. This motivates us to apply support vector machine technique to our post-stroke rehabilitation problem.

3 Hardware Design and Implementation

Figure 1 shows the hardware layout for all team members. 6-axis IMU sensor is utilized to collect kinematic information of the user during exercises. 2.4 GHz wireless communication antenna is utilised to transmit and send data to GUI. The Arduino Nano 33 BLE sense has incorporated all IMU and BLE modules as the microcontroller. Extra modules, such as SD card reader, along with push button and LED are used for data collection and hardware control. All components are powered by 5V power bank with 2.1A current output.



(a) Alan (b) Tae (c) Irfan

Figure 1: Overview of hardware setup

3.1 From breadboard to stripboard

In Figure 2 (a), the functionality of hardware prototype is firstly tested in the breadboard. In the beginning, 3-axis accelerometer and gyroscope readings are visualised in the Arduino serial plotter. Then, the datasets can be stored into SD card reader as a csv file. In addition, BLE advertised data can be transmitted to GUI. Finally, the functionality of button control and LED indication are verified before building the stripboard in the Figure 2 (b).

Several modification are taken in the stripboard design. All components are soldered in the compact stripboard, where the original jumper wires are replaced with colourful solid wires to reduce the space usage and ensure stability of connection. The microcontroller is located in the left corner for all members design to ensure the direction of sensor readings consistent. Furthermore, plastic or nylon wires are tied in each corners to attach the stripboard on the wrist band. This wrist band is ready for data collection.

(a) (b)

Figure 2: The overview of breadboard (a) and stripboard (b) layout

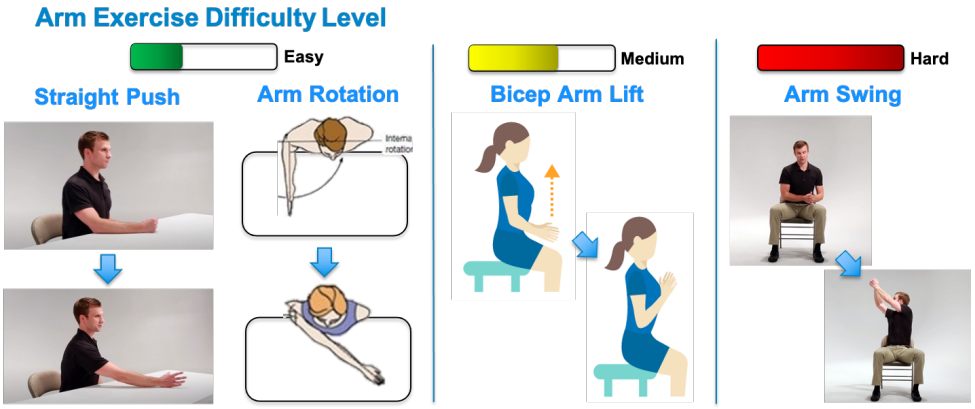
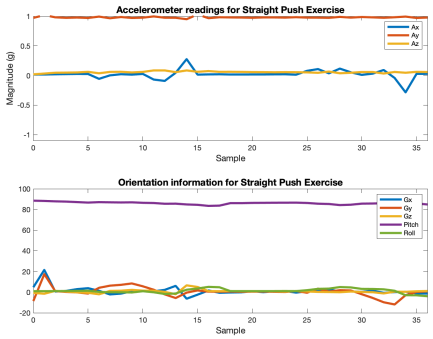


Figure 3: Arm exercises type based on difficulty level.

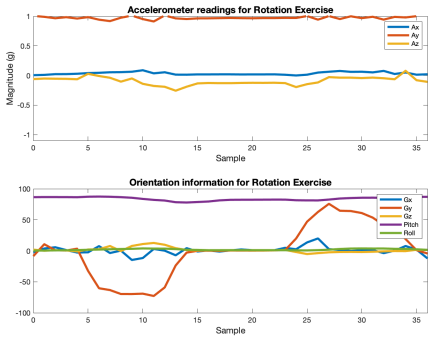
4 Data Collection

4.1 Experiment Design

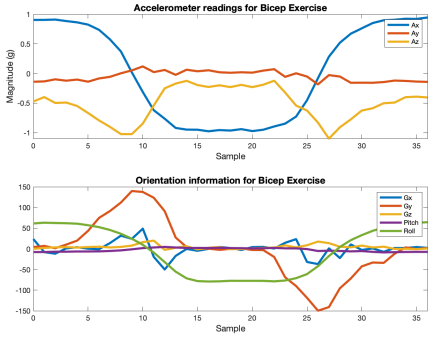
There are 4 main types of arm exercises chosen for this project. Those are divided into 3 categories based on difficulty level, i.e. easy level which are *Straight Push* and *Arm Rotation*, medium level which is *Bicep Arm Lift*, and hard level which is *Arm Swing*.



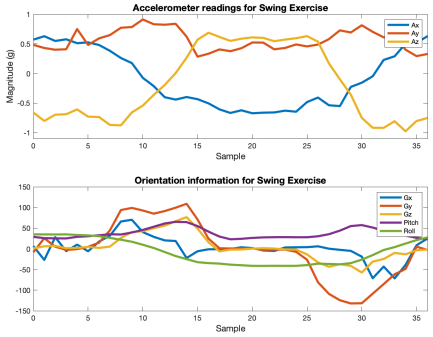
(a) Straight Push.



(b) Arm Rotation.



(c) Bicep Lift.



(d) Arm Swing

Figure 4: IMU raw data reading for arm exercises.

As seen in Figure 3, Straight Push is performed by putting the arm on the table in front of the chest with the wrist position perpendicular to the table, pushing forward the arm to the front, and pulling it backward to return. Arm Rotation is performed by putting the arm on the table, rotating the arm internally by 90° clockwise away from the chest, and rotating back by 90° counterclockwise to return. Bicep Arm Lift is performed by putting the arm to resting position on the right thigh with wrist sensor pointing downward, lifting the arm towards the shoulder, and putting it down to return. Arm Swing is performed by putting the right arm in resting position on the left thigh, swinging it towards the above right shoulder with a help from the left hand, and swinging it back to return. All these exercises are performed based on the assumption that the right arm is the affected arm.

The IMU sensor raw data reading of these 4 exercises can be seen in Figure 4. Each exercise has different patterns corresponding to the 6-axis of IMU sensor reading in time series domain. However, it is noticeable that there are some similarities in the significant axis readings between some exercises, i.e. large negative values in gx axis between arm rotation and bicep arm lift. Hence, we realised that it is quite challenging to only rely on the 6-axis IMU data to clearly identify each type of the exercises, without accompanying any important features. In the next sections, the use of these features will be explained for further process into machine learning part.

4.2 Offline Collection

Rotation angle, such as pitch and roll, are important features to classify the arm exercises with different orientation. In order to calculate reliable angle, the digital filtering is applied to low pass high-frequency motion noise. After data processing, all experiment datasets are collected in a SD card for box plot analysis. According to box plot analysis, failures of movement in each exercise are developed.

4.2.1 Data Filtering

Moving average filter with window length four is applied to smooth motion noise in the real-time acceleration data. This filter needs to wait for first four elements to be filled, which introduces around quarter second delay.

In Figure 5, the unwanted peaks in original accelerometer data (blue) are filtered out by a designed moving average filter, resulting in much smoother signal (red).

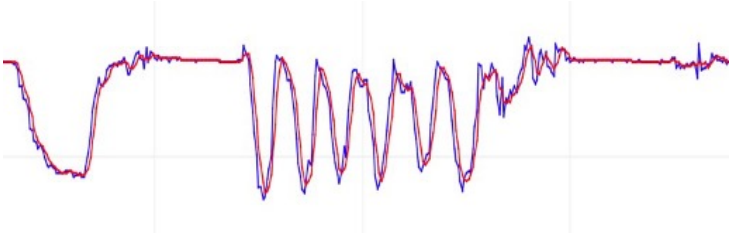


Figure 5: The 1-axis accelerometer reading with filtering

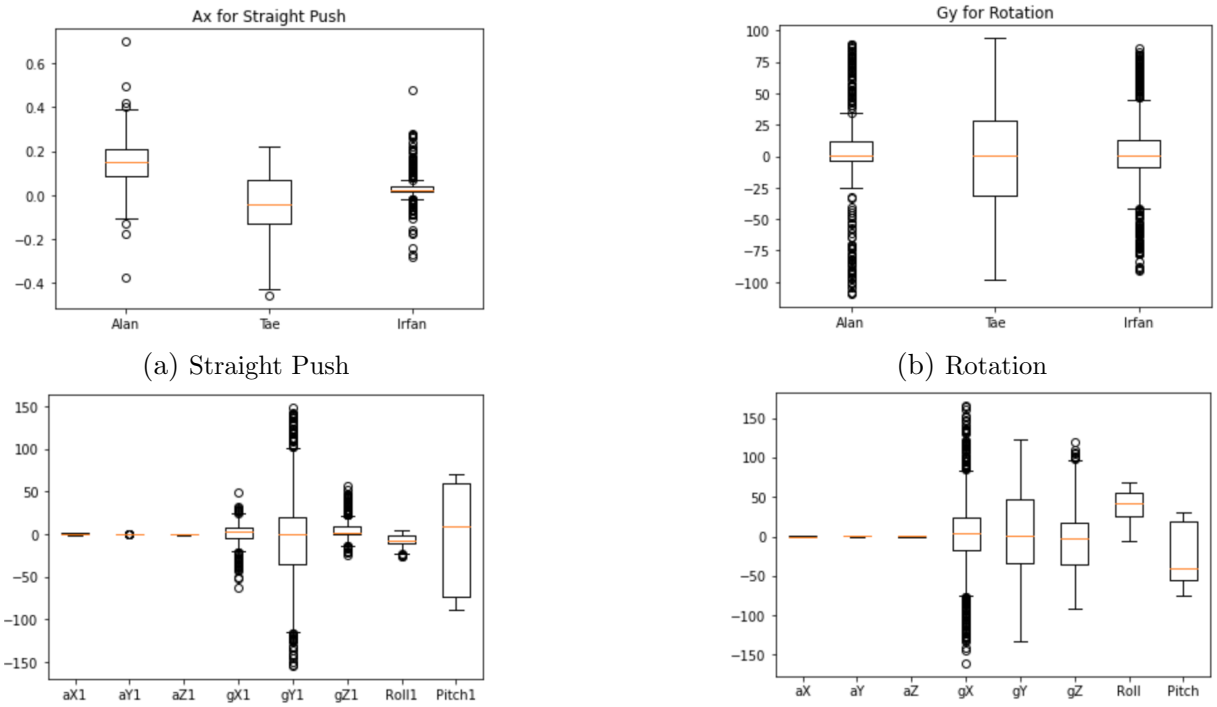
4.2.2 Euler Angle

To calculate Euler angle, either 3-axis accelerometer or 3-axis gyroscope readings can be used. The accelerometer suffers from high-frequency motion noise and the gyroscope drifts in a long term. The complex techniques, such as kalman filter, are required to remove the drift effect. In terms of filter complexity, the accelerometer is chosen since a simple low pass filter can remove the noise. The pitch and roll angles can be measured in the Equation 1 and 2 respectively.

$$Pitch = \frac{180}{\pi} * atan2(\frac{-A_x}{\sqrt{A_y^2 + A_z^2}}) \tag{1}$$

$$Roll = \frac{180}{\pi} * atan2(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}) \tag{2}$$

4.2.3 Data Analysis



(a) Straight Push

(b) Rotation

(c) Bicep

(d) Swing

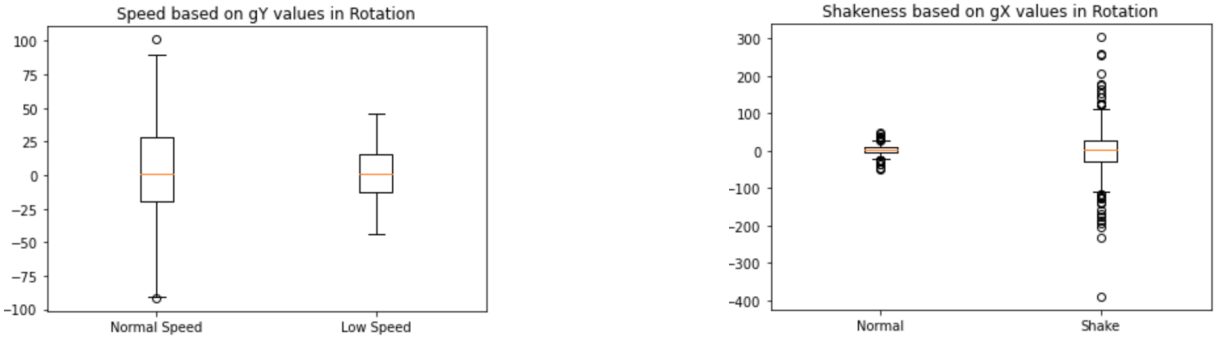
Figure 6: The boxplot analysis for 4 types of exercises

The sequence of experiment data are analysed using the box plots, which provides the statistical difference. In Figure 6 (a) and (b), the deterministic features Ax and Gy are crucial to classify the straight push and rotation exercise respectively, where the statistic range in both features are similar in all members. However, other features, like roll and pitch, are important for the rest of complex exercises, such as bicep and arm swing. As shown in Figure 6 (c), Gy and pitch angle are changed during bicep rotation, where around $\pm 90°$ pitch angle is observed. In the Figure 6 (d), 3-axis gyroscope, roll, and

pitch readings are fluctuated randomly in the arm swing exercise. All these deterministic features can be used later to classify 4 types of arm exercises.

Many outliers are observed in the collected exercise data, which confirms the variability of data in the same exercise. The designed ML model later should target these outliers to give good performance.

4.3 Failure of Movement



(a) Rotation FL

(b) Rotation FS

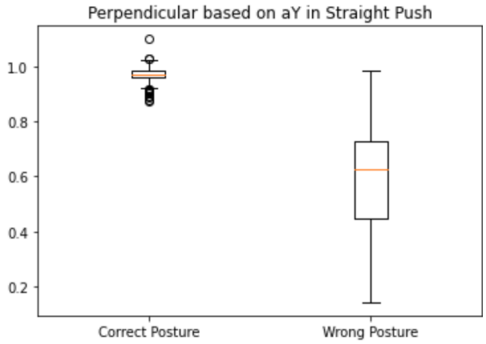
Figure 7: Failure cases of rotation due to low speed (a) and shaking (b)

Instead of identifying 4 types of exercises correctly, many failure cases are also investigated to enhance the system complexity and also provide some useful feedback.

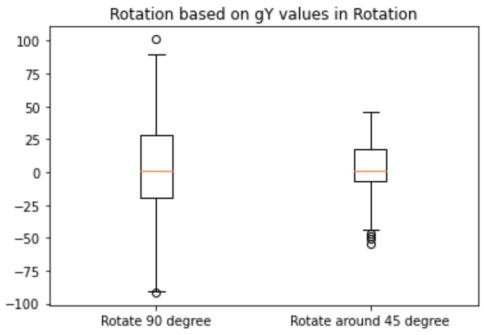
According to data analysis in Section 4.2.3 and literature review, random shaking and low speed are two common failure scenarios observed in stroke patients, which affects largely in sensor readings. For example, the magnitude range of Gy in rotation with low speed is half the normal situation and the fluctuation range of Gx in shaking rotation is double that of correct rotation shown in Figure 7. The similar statistic differences are also preserved in the other three exercises. Therefore, the low speed and shaking failures are considered for all types of exercises.

Except for commonly shaking and low speed, each exercise also has their specific failure cases to target the loss of coordination. In Figure 8 (a), wrist without being perpendicular to the table in the straight push exercise is considered as the failure, where the Ay fluctuates between 0.2 and 1. Large changes in Gy are observed in the correct rotation exercise whereas less rotation ($< 60°$) will lead to a small range of Gy shown in Figure 8 (b). Compared with the correct exercises, pitch and roll angle are significantly different in the Bicep and Arm swing failure cases (Bicep fail full cycle and swing fail $< 60°$) shown in (c) and (d).

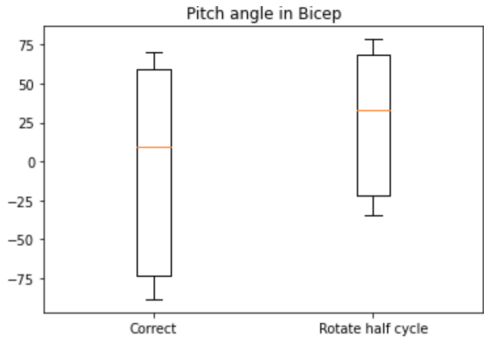
Overall, 4 types of exercises with 2 common (shake and low speed) and one specific failure cases sum up to 16 classified outputs. To remove any bias for ML model training and guarantee reliable accuracy, 100 datasets of each output are collected equally by all members, resulting in totally 1600 datasets.



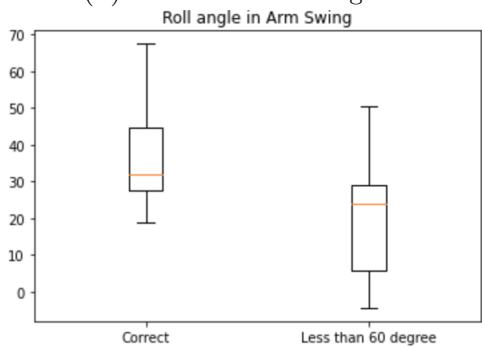
(a) Straight push fail perpendicular



(b) Rotation fail 90 degree



(c) Bicep fail full cycle



(d) Swing fail 60 degree

Figure 8: The boxplot analysis for 4 types of failure exercises.

5 Methods

Initially, all collected data are processed to train the ML models. To classify total 16 outputs, four popular models, including KNN, MLP, SVM and DT, are investigated. Standard ML procedures are taken to train, validate and test the models, which are later converted and depolyed to Arduino for further testing. The wireless BLE communication is set up to remove any hindered wire connection and provide the user with flexibility to carry out different exercises. Finally, a *state-of-the-art* GUI is developed in a `p5.js` web editor to enable the user to visualise the updated feedback and their performance analysis in two separate pages, including Home and Data visualisations. The GUI user guide is provided in Appendix B.

5.1 ML model training, validation and testing

Figure 9 illustrates the standard procedure to train and validate our ML models. Initally, the total 1600 datasets are spilt into training data (70%) and test data (30%). For hyperparameter tuning, 20% training data are split for validation data. The sequence of collected datasets with sensor readings are converted to the statistical features (mean, median, max, min and std) as ML inputs and also are annotated as corresponding exercises, resulting in totally 40 input features as a feature vector with 1 target label. With the use of encoder, the string target labels are converted to the binary matrix for neural network training and integer values for standard ML models training. Feature selection

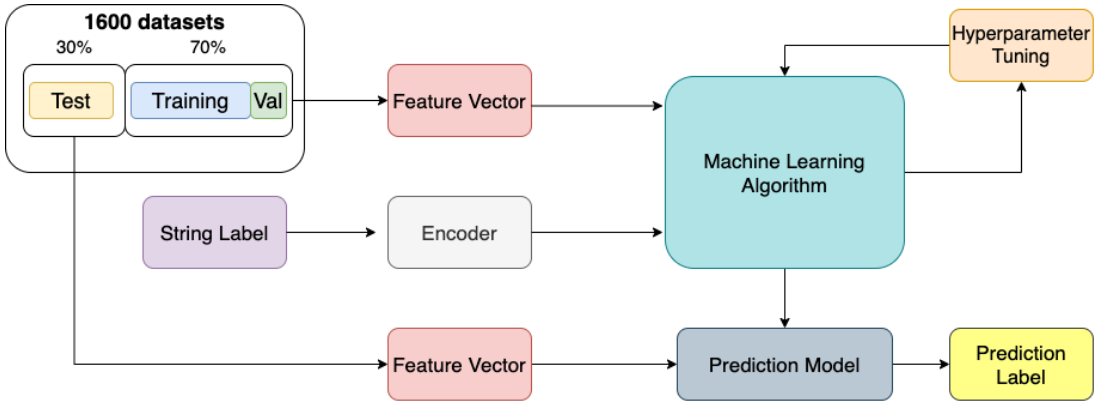
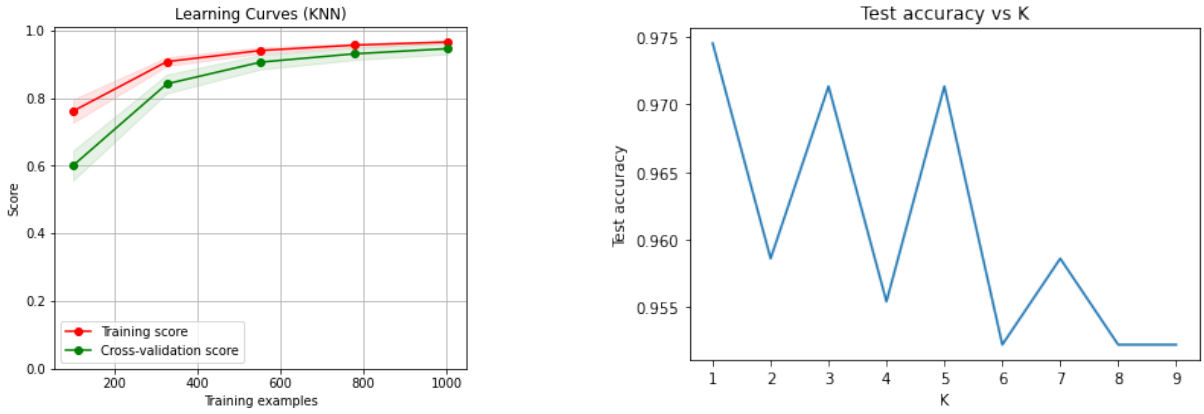


Figure 9: The overview of ML model training procedure

and scaling are employed to the input features. According to the correlation matrix in Appendix A.1, 0.985 correlation threshold is set to reduce the input dimension to 33. Due to different ranges between input features, it is suggested that standardization can convert them to normally distributed data.

Four types of ML models are investigated to classify 16 exercise outputs, including KNN, SVM, DT and MLP. Specific hyperparameters in each models are tuned on the validation sets to obtain the optimal hyperparamters. Finally, we retrain models with the optimal hyperparameters for performance analysis on unseen test data.

5.1.1 k-nearest neighbors (KNN)



(a) Learning curve

(b) Validation accuracy of different values of K

Figure 10: KNN model learning curve and validation accuracy of different K values

Figure 10 (a) shows the learning curve of KNN which indicates that the performance of the model can be improved by using more training examples. Furthermore, several runs with the different number of K are also investigated to find the optimal value of K which yields the best performance and does not overfit training data. Figure 10 (b) shows the test accuracy against the different number of K {1..9} used in KNN model. According to the plot, even though there is no significant difference between K values

and the validation performance, K = 6 is chosen as the optimal number of K, as this value corresponds to the elbow of the plot where the test accuracy starts to flatten out. Moreover, by using K=6, overfitting is less likely to occur, since the model will become more stable and robust to outliers. In this experiment, the final KNN model with the optimal value of K achieves validation accuracy 95.22%.

5.1.2 Support Vector Machines (SVM)

There are four types of *kernel functions* implemented for this project, such as Linear, Polynomial, Radial Basis Function (RBF), and Sigmoid, according to [11].

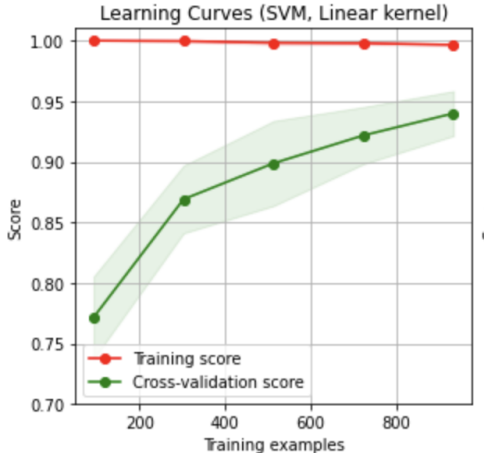


Figure 11: Learning curve plot of SVM linear kernel model.

Figure 11 shows a learning curve of SVM linear kernel between training and cross-validation score. The plot is generated using a method from [12]. It can bee seen that there is quite a gap between training and validation score. However, the gap could be decreased by having more training samples data [12].

Figure 21a, 21b, and 21c in Appendix A.2 show validation curves of SVM hyperparameter tuning of gamma. The range of gamma value experimented is {0.0001,...,0.01} and C value is {0.0001,...,10}. It is not applicable for linear kernel as it shows constant value. For polynomial kernel, the larger gamma value is better, thus using 0.001 is optimal with minimum gap between training and validation scores. For RBF kernel, gamma value is an important factor to the performance, thus choosing 0.0001 is optimal before it starts to overfit.

Figure 21d, 21e, and 21f show validation curves of SVM hyperparameter tuning of C. The C parameter controls how much we want to punish our model for each misclassified point, as it is the regularisation parameter [13]. For all kernels, the value $C = 1$ as default is already optimal with maximum validation scores at around 94%.

5.1.3 Decision Tree (DT)

The choice of tree depth has influence on the model capacity and are investigated by the validation data shown in Figure 12. As the depth increases, both accuracy increase and finally converge after depth = 6. Based on the observation, the tree depth = 6 is chosen

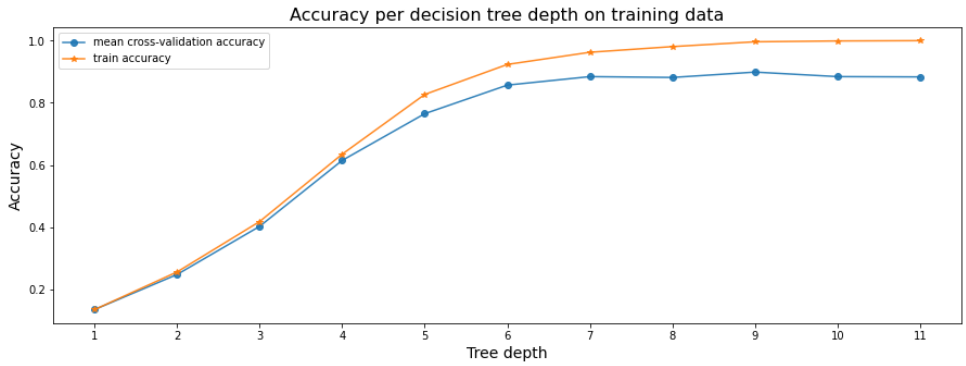
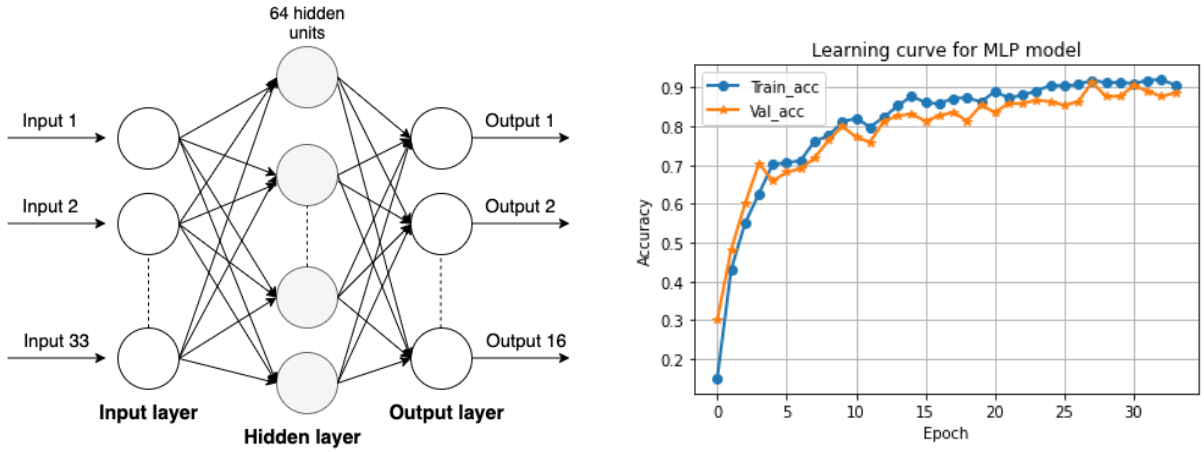


Figure 12: The accuracy of DT model with tree depth

to train the final model. Given the datasets, the entropy criterion is applied as the quality measure, where the optimal split is chosen by the feature with less entropy. We finally achieve around 92% and 88% in the trained and validation accuracy respectively.

5.1.4 Multilayer Perceptron (MLP)



(a) architecture (b) accuracy

Figure 13: The performance of the MLP architecture (a) on trained and validated data (b)

For structural optimization, number of neurons {32,64,100} and hidden layers {1,2} are evaluated, where the MLP architecture with 64 neurons in one hidden layer processed by the relu function shown in Figure 13 (a) achieves the best performance and generalize well on validation sets. After defining the model, the range of batch sizes {10, 16, 32, 64, 80, 100} and epochs {10, 50, 100} are evaluated using GridSearchCV, where the best combination is 10 batch size and 100 epochs. After compiling the Adam optimiser and categorical crossentropy loss function, 16 possible outputs are produced by the softmax activation function in the output layer. Early stop function is used to monitor the validation accuracy with patience = 6. Finally, the trained and validation data are fitted to the model to give the validation and train accuracy shown in Figure 13 (b), where the model can achieves around 90% in both accuracy after 30 epochs.

5.1.5 Model Test Performance

Type of ML model	Test Acc	Precision	Recall	F1 score
MLP	0.85	0.88	0.85	0.85
DT	0.88	0.89	0.88	0.88
KNN	0.95	0.96	0.95	0.95
SVM (RBF kernel)	0.90	0.92	0.90	0.90
SVM (polynomial kernel)	0.95	0.96	0.95	0.95
SVM (linear kernel)	0.96	0.96	0.96	0.96

Table 1: Test performance analysis for 4 types of model

Table 1 demonstrates the performance analysis of 4 best trained ML models on 471 unseen test data. Overall, all models achieve high test accuracy, precision, recall and F1 score. DT can perform slightly better than MLP but the MLP model offers more flexibility in the architecture optimization. Although KNN achieves similar performance like the SVM model, the distance-based method relies on the feature prepossessing to reduce the input dimension for less computation and memory usage, which makes them harder to implement in the small memory Arduino. In three kernels of the SVM model, the linear SVM achieves the best performance in all metrics.

In terms of flexibility and accuracy, the SVM and MLP trained model are converted and implemented in Arduino for real-scenario testing.

5.2 ML model testing in Arduino

5.2.1 Tensorflow Lite for MLP

Tensorflow lite is a popular platform to convert neural network architecture to the suitable format, which can be implemented in Arduino. Therefore, the trained MLP model is converted to C++ file in terms of byte arrays in Jupyter Notebook.

Then, the model can be loaded in Arduino with the intial setup of interpreter and resolver for testing. All statistical features are calculated and fed into the input tensor to compute the output. The updated output can be retrieved by invoking the configured interpreter. However, the testing results are generally poor with only 50% accuracy, and some exercises, like swing cannot be even detected.

The possible reasons can be that the Tensorflow lite are normally used to operate small dimension of data and large dimension in this case will lead to model failure. Also, the converted model are large byte arrays, which makes it harder to debug and understand how it operates inside Arduino system.

5.2.2 SVM Linear Kernel

Figure 14 shows a screenshot of SVM model tested on Arduino with the results displayed in Serial monitor. The display shows firstly calculation results for 40 features of each exercise. In the bottom part, the result of classification algorithm is displayed. The SVM linear kernel model is converted into C code using external library called `MicroMLgen` [14] [15] such that the model can be read as a library for code execution.

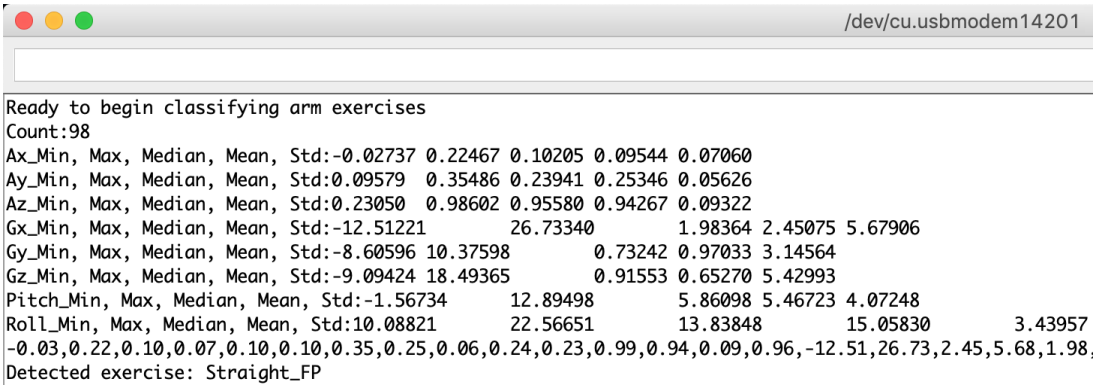


Figure 14: A screenshot of SVM model test in Arduino using Serial Monitor.

With 16 classes, there are 120 numbers of classifiers for *one-vs-one* multi-class classification [16] retrieved from the formula $n*(n-1)/2$. In one-vs-one approach, each classifier separates points of two different classes and comprising all one-vs-one classifiers leads to a multiclass classifier [11]. These classifiers create decision array and voting functions with majority votes for each output label. To provide a consistent interface with other classifiers, the `decision_function_shape` option allows to monotonically transform the results of the "one-versus-one" classifiers to a "one-vs-rest" decision function of shape `(n_samples, n_classes)` [16]. For linear kernel, the model computes 349 kernels array to predict class features vector and compute support vectors. Meanwhile, for polynomial kernel of degree 3, the algorithm produces 341 kernels array.

5.3 Graphical User Interface (GUI)

5.3.1 BLE communication

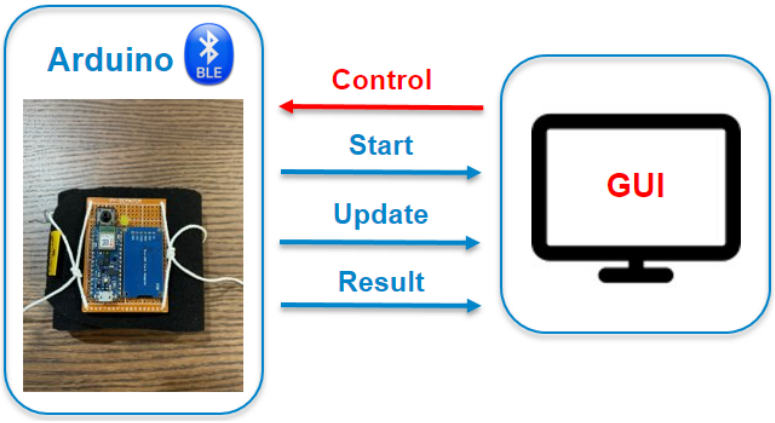


Figure 15: Signals used in BLE communication

To increase the practicality of our application and enrich user's experience by providing constructive feedback, BLE was considered and has been incorporated into our application. Figure 15 shows 4 main signals used to establish communication between

Arduino and our GUI. The first one is *controlling signal* which is used by GUI to control the starting and ending period of exercises. The second one is *starting signal* that is activated and transmitted from Arduino to GUI when a physical button on Arduino is pressed, activating the countdown timer. Furthermore, when users perform the same exercises, *updating signal* will be sent to GUI to update and increment the number of exercises performed by users. This enables our web to keep track of the total number of performed exercises. Lastly, the Arduino transmits *result signal* back to the GUI in form of classification results which will be utilized on our GUI to visualize the performance and analyze results.

5.3.2 Home Page

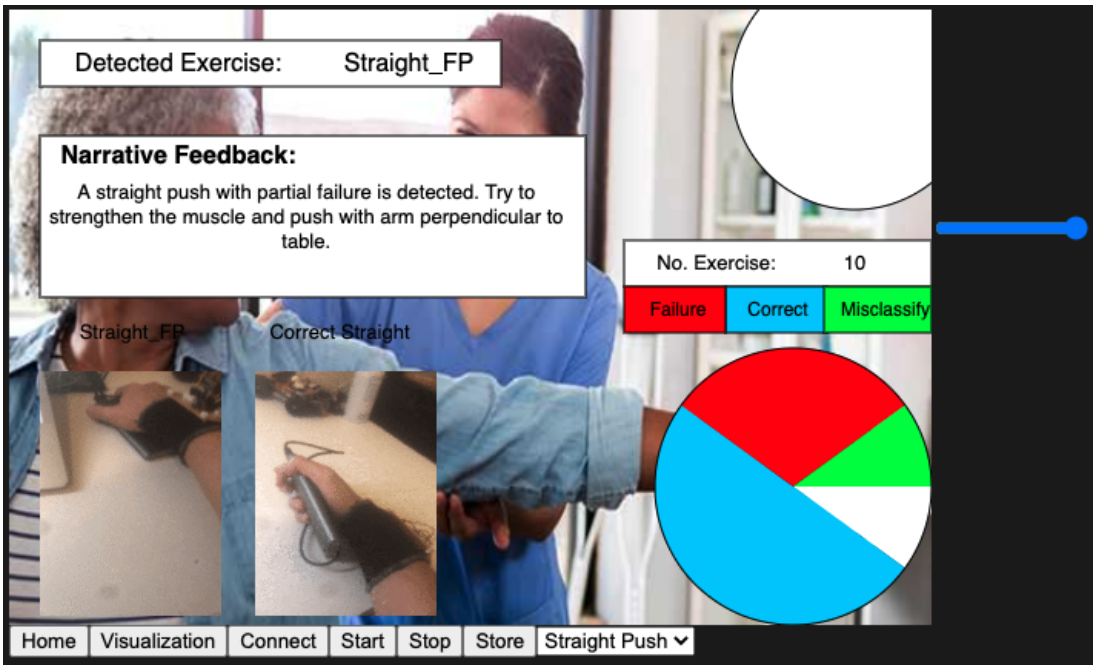


Figure 16: Home Page of GUI.

Figure 16 demonstrates the layout for the Home page. The connect button is pressed to scan the nearby advertised BLE peripherals. Additional start button is used to control when to record the exercise with the countdown timer (6 second counter). When the stop button is pressed, the classification result will be sent from Arduino and then be displayed in the Detected Exercise block with simultaneous text and animated gif images as the feedback instruction. Besides, the user can select which and how many exercises to do by selecting the option in a dropdown menu and moving a slide bar. According to the number and type of detected exercises, the pie chart will output 3 types of results, such as Failure, Correct and Misclassify. When the Store button is pressed, all exercise results can be stored into the local server and then be loaded later for the performance analysis.

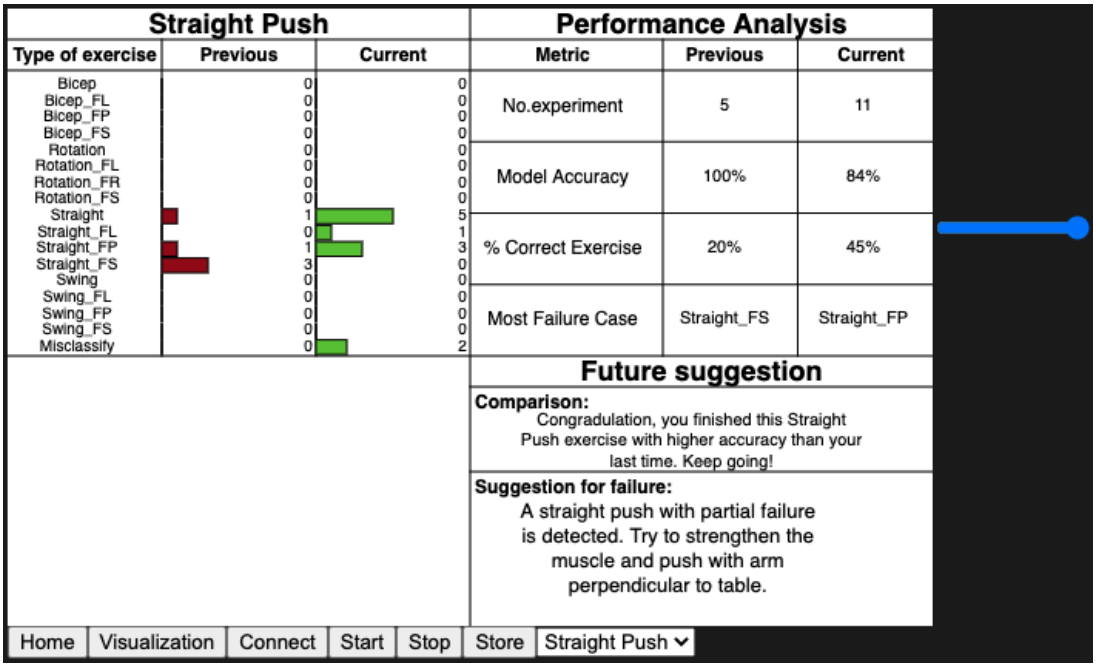


Figure 17: Data Visualization Page of GUI.

5.3.3 Data Visualisation Page

There are 3 main parts of data visualisation page, as shown in Figure 17. The first part is the bar plots that shows the comparison of the exercise counting of the chosen exercise type between the previous record and the current performance result. The previous record is shown using load function. The second part is the *Performance Analysis* that consists of 4 performance metrics, such as number of exercise movements performed by the user, model accuracy that shows how accurate ML model performs in classifying the correct chosen exercise type against miss-classifying other exercise type, the percentage of correct exercise performed by the user against any failure cases, and what the most failure case performed by the user is. The last part is *Future Suggestion* that shows some narrative feedback according to the comparison of previous record and the current performance result, and feedback for current most failure case to improve on next time.

6 Experimental Results

Characteristics	Jingchao	Irfan	Chatchanan	Subject
Sex	Male	Male	Male	Female
Age (years)	22	32	25	34
Weight (kg)	65	55	66	54
Height (cm)	175	160	169	168
BMI (kg/cm^2)	21.2	21.5	23.1	19.1

Table 2: Subject's characteristics

6.1 Test Results

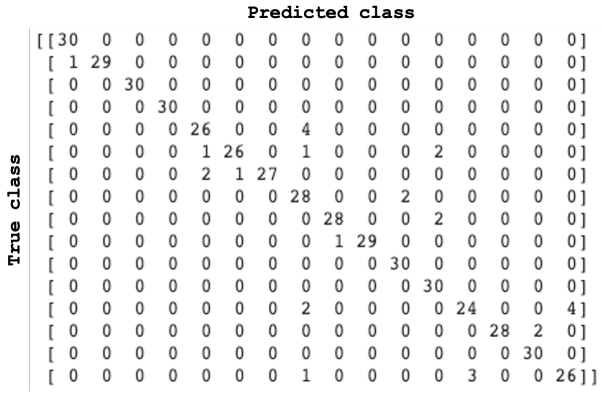


Figure 18: Confusion Matrix of Test Results within team members.

All exercises are performed within the instruction discussed in the Section 4.1. Table 2 shows the subject's characteristics for experiment testing. Figure 18 shows a confusion matrix of real-scenario test results of ML model in Arduino with 10 trials per exercise class for each team member, hence 480 data in total.

In addition, one external person was invited to perform all exercises in order to validate the reliability of the device. In this experiment, the subject performs 5 trials per exercise and was informed of information about each exercise, resulting totally 80 data samples.

6.2 Discussion of results

According to Figure 18, we achieve 94% accuracy in the real-life experiment testing. Since our exercises vary in each time, the model accuracy indicates satisfactory robustness of our trained linear SVM model in the exercise classification. Several misclassified cases shown in the matrix are during the rotation exercises, where the correct rotation is sometimes detected as the shaking rotation. The possible reason is that the environment (coarse table) restricts the rotational orientation, which introduces some unwanted shaking. Overall, except for shaking cases, all the exercises are detected correctly by our model. The high accuracy cannot be achieved without the feature analysis in Section 4.3 and 4.2.3, and the ML model training procedure in Section 5.1.

To further verify the viability of whole system, the unseen data distribution is investigated. A healthy female subject is invited to test the system, where the ML model can only achieve around 84% classification accuracy. The brief guidance of movement is provided in the beginning but the subject can do flexibly during exercises within the instruction, which potentially degrades the model performance. Also, each subject may do the exercises differently. The possible solution for the poor accuracy is to collect more datasets from different people with various distribution on the same exercises.

Overall, the system at this stage is still robust to achieve promising performance in the experiment testing by our team and a subject.

7 Workplan

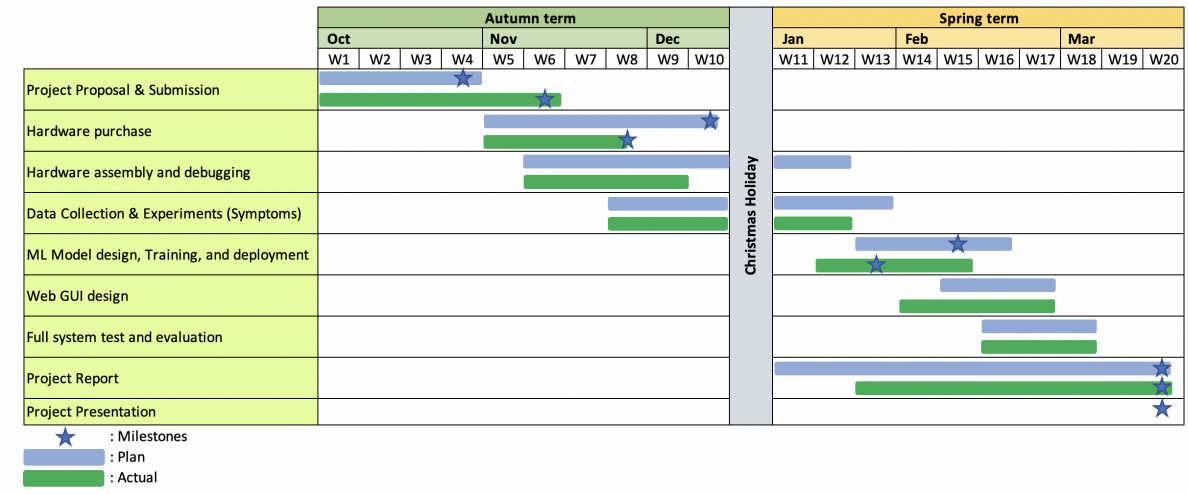


Figure 19: Gantt Chart Timeline for Workplan of this project.

Figure 19 shows a gantt chart timeline for the workplan of this project, including the actual timeline realisation. At the beginning stage of this project, we were a bit behind schedule in terms of project proposal submission, due to some changes in circumstances for the project ideas, including back and forth communications with the supervisors to get approval of the project topic. However, we were able to catch up on this delay by having a shorter period of hardware purchase, assembly and debugging. Data collections, experiments, and machine learning model design and training stages were also performed in relatively shorter period than planned. Meanwhile, the Web GUI completion period was a bit longer than predicted. Nevertheless, we were still able to achieve completion of this project on time at the end for the final presentation and report submission.

8 Conclusion

8.1 Summary

To address the problem of automatically classifying and assessing the performance of post stroke rehabilitation exercise, the ML-based device has been proposed to classify 4 types of arm exercises. Several steps of hardware development starting from breadboard prototype to final hardware assembly have been achieved successfully. After data preprocessing, 6-axis IMU and Euler angle are collected by each team member to initially visualize the experimental data and to analyze important features. Later, the feature extraction procedure are performed on the collected data in order to form our feature vectors. The standard ML procedure is applied to train, validate and test ML models. KNN, SVM, DT and MLP are thoroughly investigated and considered for misclassication task. Linear SVM is chosen as our choice of model to implement in the Arduino.

Moreover, real-life device testing is carried out by each member to validate the performance of the model. Additionally, to provide users with automatic feedback and facilitate

the ease use of our device, BLE communication was established to make data transmission between the device and any PCs possible. GUI has been developed into our system to offer user-friendly interface to users. Lastly, the whole system is tested and evaluated to spot potential flaws which can be used to further improve our application in the future.

8.2 Evaluation

Due to the COVID-19 situation, we cannot ask and invite some stroke patients to test and verify our system. Apart from it, we achieve all the objectives listed in Section 1.2. To tackle the viability of our system, we read several papers targeting on the stroke symptom and arm recovery. Based on the research, we add some variability to our collected datasets by changing the environment (different tables etc.) and doing exercises flexibly to better simulate the patient's possible behaviour. Moreover, to provide some useful feedback on GUI, many possible failure cases of different exercises are taken into account according to the data analysis in Section 4.3. Finally, a friendly platform is tested and implemented in Arduino wirelessly with high classification accuracy.

This project cannot work remotely without regular meetings and effective team management. Except for twice meetings a week, we make the use of several platforms, such as Github and Trello, to share, comment and discuss our work.

8.3 Future Work

Interestingly, the wireless system degrades the classification accuracy of ML model compared with the wired system. Since we use buffers to calculate features in Arduino, the possible reason is that the transmitted data in a BLE communication interrupt the running buffers, which misses some important sensor readings and leads to wrong input features calculation. The viability of system should be further verified and improved by stroke patients and another clinical experts. Below are some potential developments for future work of this project:

- Improve testing performance of ML model in the wireless BLE communication.
- Collect more datasets from different people to improve the robustness of ML model.
- Add more functionalities of GUI to provide more useful feedback.
- Ask and invite clinial experts and patients to test the whole system.

References

[1] A. Kaku, A. Parnandi, A. Venkatesan, N. Pandit, H. Schambra, and C. Fernandez-Granda, "Towards data-driven stroke rehabilitation via wearable sensors and deep learning," in *Machine Learning for Healthcare Conference*. PMLR, 2020, pp. 143–171.

[2] K. S. Yew and E. Cheng, "Acute stroke diagnosis," *American family physician*, vol. 80, no. 1, p. 33, 2009.

[3] M. Rensink, M. Schuurmans, E. Lindeman, and T. Hafsteinsdottir, "Task-oriented training in rehabilitation after stroke: systematic review," *Journal of advanced nursing*, vol. 65, no. 4, pp. 737–754, 2009.

[4] D. M. Smith, S. L. Brown, and P. A. Ubel, "Mispredictions and misrecollections: challenges for subjective outcome measurement," *Disability and Rehabilitation*, vol. 30, no. 6, pp. 418–424, 2008.

[5] M. H. Lee, D. P. Siewiorek, A. Smailagic, A. Bernardino, and S. B. i. Badia, "Learning to assess the quality of stroke rehabilitation exercises," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 218–228.

[6] S. Das, L. Trutoiu, A. Murai, D. Alcindor, M. Oh, F. De la Torre, and J. Hodgins, "Quantitative measurement of motor symptoms in parkinson's disease: A study with full-body motion capture data," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2011, pp. 6789–6792.

[7] S. Jacob, V. G. Menon, F. Al-Turjman, P. Vinoj, and L. Mostarda, "Artificial muscle intelligence system with deep learning for post-stroke assistance and rehabilitation," *IEEE Access*, vol. 7, pp. 133 463–133 473, 2019.

[8] M. Panwar, D. Biswas, H. Bajaj, M. Jöbges, R. Turk, K. Maharatna, and A. Acharyya, "Rehab-net: Deep learning framework for arm movement classification using wearable sensors for stroke rehabilitation," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 11, pp. 3026–3037, 2019.

[9] Y. Ge, Q. Wang, L. Wang, H. Wu, C. Peng, J. Wang, Y. Xu, G. Xiong, Y. Zhang, and Y. Yi, "Predicting post-stroke pneumonia using deep neural network approaches," *International journal of medical informatics*, vol. 132, p. 103986, 2019.

[10] T. Hamaguchi, T. Saito, M. Suzuki, T. Ishioka, Y. Tomisawa, N. Nakaya, and M. Abo, "Support vector machine-based classifier for the assessment of finger movement of stroke patients undergoing rehabilitation," *Journal of Medical and Biological Engineering*, vol. 40, no. 1, pp. 91–100, 2020.

[11] H. Marius. Multiclass classification with support vector machines (svm), dual problem and kernel functions. [Online]. Available: https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02

[12] scikit-learn developers (BSD License). Plotting learning curves. [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

[13] Felipe. Choosing c hyperparameter for svm classifiers: Examples with scikit-learn. [Online]. Available: https://queirozf.com/entries/choosing-c-hyperparameter-for-svm-classifiers-examples-with-scikit-learn

[14] Simone. How to deploy an arduino machine learning classifier in 4 easy steps. [Online]. Available: https://eloquentarduino.github.io/2019/11/how-to-train-a-classifier-in-scikit-learn/

[15] ——. Introducing micromlgen. [Online]. Available: https://github.com/eloquentarduino/micromlgen

[16] scikit-learn developers (BSD License). 1.4. support vector machines. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#scores-probabilities

Appendix A Model Training and Validation

A.1 Correlation matrix

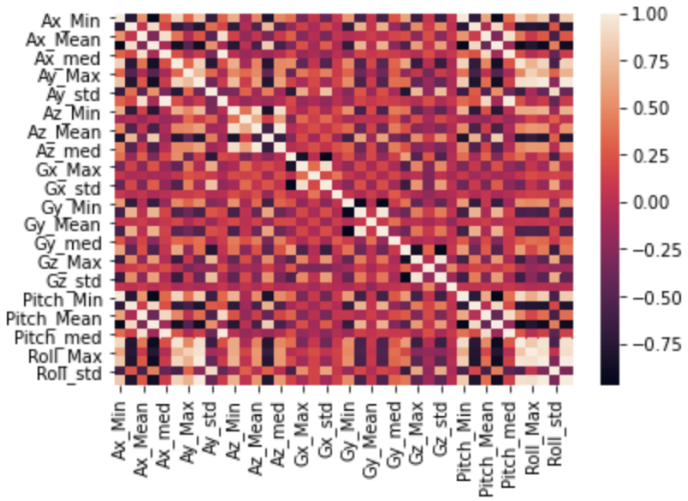
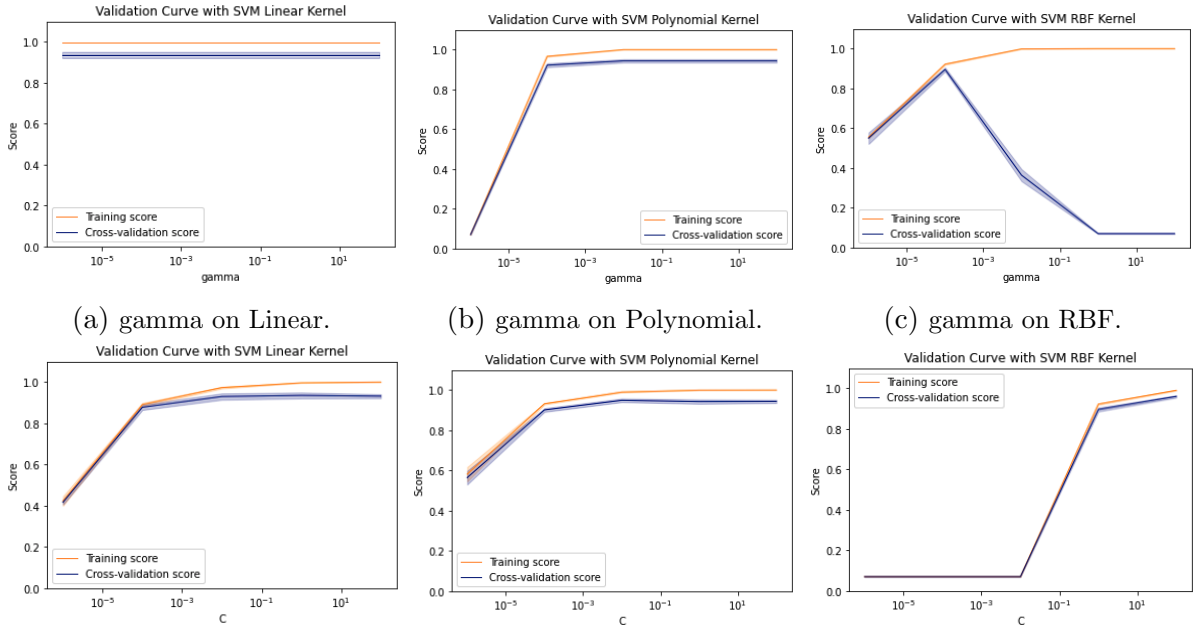


Figure 20: Correlation matrix for 40 input features

A.2 SVM



(a) gamma on Linear. (b) gamma on Polynomial. (c) gamma on RBF.

(d) C on Linear. (e) C on Polynomial. (f) C on RBF.

Figure 21: SVM model validation curves for hyperparameters tuning (gamma & C).

Appendix B GUI User Guide

The purpose of the GUI platform is for the user to analyse their performance and obtain future suggestion. User interface manual is provided for future development and testing.

B.1 Home Page

Please follow the instruction below:

- Press Connect button to scan the nearby advertised BLE devices
- Select the type of exercise you want to perform in the dropdown menu
- Select the number of exercises by moving a sliding bar
- Press the Start button or push button in hardware to record the exercise and stop for finishing exercise
- Narrative and visual feedback are given for each exercise
- After finishing each exercise, the user can select other exercises
- Finally, the local server should be connected to store all exercises in the csv file by pressing Store button

B.2 Data Visualisation Page

- According to exercise name, the user should change filename accordingly to load the exercise
- Current performance and past performance (csv file) are analysed to give performance analysis and future suggestion