**MSc Dissertation**

**Final Report**

**MSc in Integrated Machine Learning Systems**

**Student:** Svetlana Grant

**Student number:** 19132626

**Project Title:**

**Machine Learning Approach for Parkinson Disease Monitoring Using Wearable Technologies**

**Supervisor:** Prof. Izzat Darwazeh

University College London

Dept. of Electrical and Electronic Engineering

2020-21

# Table of Contents

[List of Figures 3](#_Toc81509106)

[List of Tables 4](#_Toc81509107)

[Abstract 5](#_Toc81509108)

[Executive Summary 6](#_Toc81509109)

[1) Introduction and Problem Statement 8](#_Toc81509110)

[2) Context, Background and Literature Review 10](#_Toc81509111)

[2.1. Approaches to the analysis of wearable sensor data 11](#_Toc81509112)

[2.2. Supervised ML for PD wireless sensor data 12](#_Toc81509113)

[2.3. Unsupervised machine learning and deep learning for PD 12](#_Toc81509114)

[3) Theory / Methodology 14](#_Toc81509115)

[3.1. Research Methodology and Implementation 14](#_Toc81509116)

[3.2. Expected Outcome and Impact 19](#_Toc81509117)

[4) ML Workflow Implementation and Optimisation Discussion 20](#_Toc81509118)

[4.1. Data Engineering Discussion 20](#_Toc81509119)

[4.2. Model Engineering Discussion 21](#_Toc81509120)

[4.2.1. Convolutional Deep Learning Model Architecture and Parameters 21](#_Toc81509121)

[4.2.2. Long Short-Term Memory Recurrent Neural Network Model Architecture and Parameters 24](#_Toc81509122)

[4.2.3. Data Split into Training – Validation – Testing sets 24](#_Toc81509123)

[4.2.4. Supervised Learning Model 26](#_Toc81509124)

[4.2.5. Machine Learning Model Performance 26](#_Toc81509125)

[4.3. Model Deployment Discussion 27](#_Toc81509126)

[4.3.1. Optimisation and Compression: TensorFlow Lite 27](#_Toc81509127)

[4.3.2. Further Model Compression: TensorFlow Lite Micro 29](#_Toc81509128)

[4.4. Results Discussion 30](#_Toc81509129)

[5) Conclusions and further work 33](#_Toc81509130)

[References 36](#_Toc81509131)

[Appendices 39](#_Toc81509132)

[Appendix A – Engineering Log Book 39](#_Toc81509133)

[Appendix B – Link to GitHub Project Depository 42](#_Toc81509134)

[Appendix C - Neurologist Discussion Questionnaire 42](#_Toc81509135)

[Appendix D - Neurologist Discussion Summary 43](#_Toc81509136)

# List of Figures

[Figure 1 Summary of Machine Learning Studies for PD 9](#_Toc81508841)

[Figure 2 Wearables and Machine Learning for PD: Examples of Use Cases 9](#_Toc81508842)

[Figure 3 Machine Learning Workflow 14](#_Toc81508843)

[Figure 4 MJFF Levodopa Wearable Sensors Dataset (2015) 14](#_Toc81508844)

[Figure 5 ML For PD Study: Modelling Approach 19](#_Toc81508845)

[Figure 6 CNN Model Architecture Summary 21](#_Toc81508846)

[Figure 7 CNN Model Architecture Summary 23](#_Toc81508847)

[Figure 8 LSTM Model Architecture Summary 23](#_Toc81508848)

[Figure 9 Machine Learning Model Flow: Model Compression Step 26](#_Toc81508849)

[Figure 10 TensorFlow Model Quantization: Compressing Floating Point Numbers [36] 27](#_Toc81508850)

[Figure 11 Example of Microcontroller Memory Usage [35] 28](#_Toc81508851)

# List of Tables

[Table 1 Modelling Data Split for MJFF BOS Patient Data sets 23](#_Toc81508404)

[Table 2 Deep Learning Models: Accuracy and Size 25](#_Toc81508405)

[Table 3 TensorFlow Frameworks: Comparison of TF, TFLite and TFLite Micro 26](#_Toc81508406)

[Table 4 Machine Learning Models: Accuracy Metrics 30](#_Toc81508407)

[Table 5 Machine Learning Models: Size Metrics 31](#_Toc81508408)

# Abstract

**Background:** The use of Machine Learning algorithms for the analysis of wearable sensor data for Parkinson’s Disease (PD) has grown in importance. As number of patients with PD is on the rise, with 10 million predicted by 2030 globally, the ability to monitor the disease remotely needs to be established. In the last decade, successful application of machine learning methods to various medical and healthcare problems have led to the growing number of studies dedicated to the application of supervised, unsupervised and deep learning machine learning algorithms to PD. All demonstrated high accuracy results for remotely tracking PD symptoms, such as tremor, dyskinesia, bradykinesia and freezing of gait. Overall, deep learning models have been more accurate than simpler supervised models, however, suffered from such constraints as large model size, high computational cost and power consumption, which made them unsuitable for wearables, as well as lack of transparency that gained neural nets the reputation of a “black box”.

**Objective:** The emergence of Tiny ML, which allows to significantly compress deep learning models without losing accuracy, calls for the review of the suitability of deep neural networks for wearables. This study evaluated the application of both deep and supervised learning approaches for monitoring the PD symptom of dyskinesia. It used accelerator sensor data from the Michael J Fox Foundation (MJFF) Levodopa Wearable Sensors dataset collected in 2015.

**Results:** The study explored whether the application of TinyML methods, such as quantization (reformatting data from float32 format to int8 format) and pruning (reducing the number of inputs), to deep neural networks, can improve their overall robustness and make their deployment onto embedded PD wearables possible. It established that significant compression to below 20KB can be achieved without compromising overall accuracy, however, identified small size of data and class imbalance as the new areas of exploration.

# Executive Summary

Definition: Parkinson’s Disease (PD) is a chronic neurological disease, which affects over 5 million people worldwide, with more than double of this number expected by 2030 [1]. Tracking PD patients’ gait by means of wearable sensors is an important part of remoting monitoring for Parkinson’s Disease, helping to diagnose it, track its progression, and ascertain how a patient reacts to the medicines in their daily life.

Tracking some PD symptoms is now possible on commercial wrist – wearable devices, for example, a Motor Fluctuations Monitor for Parkinson’s Disease (MM4PD) app was launched in 2021 for iPhone [2]. It is equally important to be able to track symptoms via patient’s gait, during different activities of daily living. Such symptoms as Dyskinesia – involuntary movements in reaction to taking Levodopa, for example - manifest themselves on patients’ legs. [3] [4]

The gap, and how this study fills it: Use of machine learning models for classification and regression has been extensively researched in the last decade. Yet, adoption of wearables and machine learning for remote monitoring of Parkinson’s Disease is still at a nascent stage, where lack of confidence in technology and acceptable integration in daily life are among the limiting factors [5]. In parallel, the field of TinyML has emerged to enable the use of machine learning on wearables and microcontrollers. This study combined the two fields: It researched ML models that can provide high levels of accuracy with a focus on deep learning models; it then investigated whether these models can be compressed to fit on microcontrollers without significant loss of accuracy.

What the study is about: The study investigated the performance of deep learning CNN and LSTM models that have shown best accuracy for time series data in previous PD research. These were used to classify dyskinesia, using the Michael J Fox Foundation (MJFF) Levodopa dataset collected in 2015. Deep learning model performance was compared to supervised machine learning Random Forest ensemble of models. Both sets of models were compressed using TensorFlow Lite framework, to conclude whether they can reach the size of under 20 KB, which is required for deployment on edge devices that use microcontrollers.

Key achievements of the study: This study combined the use of neural networks for PD wearables; and demonstrated that significant compression (x10) can be achieved for deep learning models without compromising accuracy levels, making these models suitable for deployment on edge devices that use microcontrollers.

Conclusions and next steps: Model accuracy for CNN and LSTM was achieved in the range of 97%, although only low levels of sensitivity were demonstrated without further model optimisation. The accuracy levels were maintained after compressing the models to less than 20KB. The MJFF dataset, however, is very small – only 17 patients wore Shimmer sensors on lower limbs in the MJFF Levodopa study. This means that the findings of the study would need to be extended and validated for a larger number of patients. Moreover, due to class imbalance, the prediction accuracy for smaller classes was not sufficiently high, and further optimisation of the models would need to be performed. Finally, further exploration into the transparency of deep learning models is needed, to continue building PD practitioner and patient confidence in ML. [[1]](#footnote-1)

# Introduction and Problem Statement

Parkinson’s Disease is a chronic neurological disease, which affects over 5 million people worldwide, with more than double of this number expected by 2030 [1]. In the last decade, advances in wearable sensor technologies and growing use of machine learning (ML) for medical purposes have opened new possibilities for tracking and monitoring the progress of Parkinson’s Disease (PD). Historically, clinical neurology data for diagnosing and tracking the progress of Parkinson’s Disease has been collected on a small scale in a clinical environment, with long time gaps between patient evaluation sessions. Recently, there has been growing interest in evaluating the use of medical sensors for continuous remote monitoring, and several studies, for example, Tzallas, 2014 [6], Sigcha, 2020 [7] and Mancini, 2021 [8] have tested their use in the home environment.

Home monitoring using wearable systems can be particularly beneficial for PD patients who need to be observed and evaluated on a regular basis, who are prone to injuries due to problems with balance and suffer from such debilitating symptoms as dyskinesia (involuntary erratic movements) , bradykinesia (slowness of movement) and freezing of gait (FoG). Wearable sensors can capture continuous changes in patients’ motor and non-motor symptoms, improving accuracy and frequency of observation.

Wearable Inertial Measurement Unit (IMU) sensors with accelerometers, gyroscopes and magnetometers, as well as pressure sensors, can be used to measure various symptoms of PD, including tremors, bradykinesia and gait freezing, as well as effects of PD medicines, for example, dyskinesia [6]. In the aftermath of COVID-19, the need for remote monitoring and enablement of medical decision support systems outside of hospitals is expected to increase even further.

In parallel, use of Machine Learning algorithms for the analysis of wearable sensor data has grown in popularity. Earlier wearable sensor studies have focussed on using sensor data for PD predictions based on statistical signal processing approaches, most notably, threshold-based algorithms which allow to establish movement and gait patterns, explored in the studies by Bachlin et al, 2010 [9], Ferrari et al, 2016 [10], Tunca et al, 2017 [11], and Mancini et al, 2021 [8].

More recently, successful application of machine learning methods to medical and healthcare problems have led to the growing number of studies dedicated to the application of supervised, unsupervised and deep learning to PD. High accuracy results of using supervised and deep learning methods to Parkinson’s were demonstrated in such studies as Camps et al, 2018, [12], Sama et al, 2018 [12], Zhao et al, 2018 [13], Hssayeni et al, 2020 [3].

At the same time, the use of ML for medical wearables is still new, and the application of these systems to support medical decision support systems needs to be explored further. One of the questions that need to be answered is which machine learning approach is most suitable for different types of PD use cases. The requirements for the data need to come from the end-users: PD patients themselves who will want to know how their data helps to support medical decisions; and PD clinicians who will be on the receiving end of the data collected in a remote, unmonitored environment.

While deciding between different ML models, a choice is often presented as a trade-off between simpler but less accurate supervised models and more accurate “black box” deep learning models [14]. Both approaches have benefits and constraints discussed in this report. Among the constraints of the deep learning models are their size, high computational cost and power consumption, as well as lack of transparency. The emergence of the TinyML framework that optimises the size and performance of deep neural networks for embedded devices calls for a revisit of this trade-off.

This study aims to evaluate the application of different types of machine learning approaches to main PD use cases and explore how application of TinyML methods, such as quantization and pruning, deep learning models, combined with interpretable methods can improve their overall robustness and trustworthiness, and make their deployment onto embedded PD wearables possible. The following objectives were set for this study:

1. Compare performance of deep neural networks-- Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) -- and a supervised machine learning model – Random Forest algorithm:
   1. Compare accuracy (Sensitivity, Specificity and AUC (area under the curve, which establishes how well the model differentiates between different classes))
   2. Compare size of the trained model
2. Compress and optimise the models using Tiny ML – the area of machine learning that studies the application of ML to constrained devices, such as wearables.
3. Evaluate the losses to the accuracy after the model compression.

The study established that both deep learning and supervised models reached high levels of accuracy, with CNN showing the best results on AUC (99%) and was compressed to under 20KB using the TensorFlow Lite framework. It could not, however, accurately predict the smallest classes (dyskinesia severity levels of 3 and 4).

The study aims to contribute to the growing body of research that used deep learning for classification of symptoms in Parkinson’s Disease, combining the evaluation of these models with Tiny ML methods and aiming to facilitate the deployment of ML models on wearable devices.

The report is structured as follows: Recent research review in Chapter 5 is followed by the description of the methodology and approach in Ch.6. The implementation of the machine learning flow, including data and model engineering, are detailed in Ch.7, followed by the conclusions in Chapter 8.

# Context, Background and Literature Review

A growing body of research has demonstrated that remote monitoring for Parkinson’s Disease (PD) in combination with machine learning can deliver significant value to both medical professionals and PD patients [15], [16], [17].

This study identified 26 wearables studies that used a variety of machine learning (ML) algorithms for PD research between 2001-2021, summarised in **Fig.1**. Of these, supervised ML methods have been applied to track specific daily activities to monitor and classify tremors, bradykinesia, dyskinesia, gait freezing and other gait disturbances. The research in more recent years has focussed primarily on using deep learning for freezing of gait, classification of Unified Parkinson’s Disease Rating Scale (UPDRS) stages, and classification of dyskinesia. Other extensions of using ML for gait analysis includes establishing individual gait profiles that can be useful for other types of health and fitness analysis, as for example in this UCL study by M. Hammouda [18].

Chart, scatter chart

Description automatically generated

**Figure 1 Summary of Machine Learning Studies for PD**

To date, the wearable sensor has focused on three groups of PD use cases, which would benefit from the use of machine learning approaches, as presented in **Fig. 2**.

Graphical user interface, text

Description automatically generated

**Figure 2 Wearables and Machine Learning for PD: Examples of Use Cases**

## Approaches to the analysis of wearable sensor data

The analysis of wearable sensor PD data, which consists of time-series sensor readings, is broadly based on two types of approaches – statistical threshold-based algorithms and machine learning algorithms.

**Threshold – based algorithms** use signal processing methods, including algorithms tracking changes in signal energy, and to determine specific stages of the gait phases, such as Initial Contact (IC) and Toe-Off (TO). These algorithms have been commonly applied to the recognition of specific events, such as detection of fall and Freezing of Gait (FoG), and a wide range of gait characteristics and parameters. Examples of this approach include the Freezing of Gait research performed by Baechelin et al in 2010 using the Daphnet FoG dataset [9], which used power spectral density threshold to differentiate between walking, standing and freezing episodes. More recently, the research by Mancini et al [8] derived an algorithm combining the power threshold calculated by Fast Fourier Transform (FFT) with the correlation between right and left angular velocity.

Other research demonstrated other uses of threshold – based algorithms to derive spatio-temporal gait characteristics. Ferrari et al, 2016 [10] used zero-velocity-update gait analysis system based on Kalman filter for real – time detection gait patters. Tunca et al, 2017 [11] extended the zero-velocity update and Kalman filtering methodology to non-hospital settings, to derive a rich set of standard gait metrics. Keloth et al, 2019 [19] established the variability of gait between PD and control subjects by means of measuring left and right foot angular velocity and angle differences.

The threshold- based approach remains popular due to its computational efficiency and transparency. At the same time, it presents a challenge when statistical model of the data is unknown from the start: Baechlin et al [9] observed that use of global thresholds led to lower specificity (true negatives, or proportion of actual predicted negative cases) and sensitivity (true positives, or proportion of actual predicted positive cases), which indicated that the user – independent model did not generalise as well as user-specific model. Threshold-based algorithms also struggle with high-dimensional data that contains a large number of features, making it difficult to identify average daily living (ADL) activities.

Use of **Machine Learning algorithms** for the analysis of wearable sensor data solves some of these challenges, and their application to PD monitoring has been widely studied in the last decade. Kubota et all [20] reviewed 17 wearables studies that used a variety of machine learning algorithms for PD research between 2001 and 2016. Since then, the research by Camps et al, 2018 [12] and Sigcha et al, 2020 [7] demonstrated that machine learning had better performance for FoG event detection, with sensitivity rates in excess of 90% compared to 85% rates achieved with the best threshold algorithms [8]. Sigcha et al [7] noted that “The proposed [Support Vector Machines] data representation presents advantages over previous handmade feature extraction methodologies and shows opportunities for the improvement of FOG detection systems to be applied in real time**.**”

In summary, machine learning extends traditional statistical methods, such as “parametric and nonparametric null hypothesis testing, linear and logistic regression, discriminant analysis, principal components, factor analysis, and cluster analysis”, “to cope with high dimensionality and nonlinearity, which is of particular importance in wearable sensor data.” [20].

## Supervised ML for PD wireless sensor data

As summarised in **Fig. 1**, the bulk of the ML research for PD conducted during 2009-2018 has used supervised learning algorithms for PD classification, such as SVM (Support Vector Machines), decision trees and random forest.

Supervised learning uses an algorithm that identifies a relationship between input data and output data, using the labels where “each training input must be associated with an output value” [20]. An output data represents either continuous set of values, which uses regression analysis, or a finite set of discrete data, which uses logistics regression analysis. In case of PD, both regression and classification can apply, depending on the problem that is being solved. Prediction of an event (for example, FoG or dyskinesia) will apply binary classification, prediction of activity will use multi-label classification approach, while UPDRS score can use either multi-label classification or a regression to derive a numeric score. While traditional statistical regression analysis is typically linear, supervised ML allows to establish non-linear, high-dimensional relationships.

However, labelling data collected from wearable sensors requires a complex set-up, typically performed in a hospital environment, when either a clinician or a patient validates an event. Thus, most data collected from wearable sensors is unlabelled, which has prompted the use of unsupervised machine learning and deep learning techniques.

## Unsupervised machine learning and deep learning for PD

Unsupervised learning is establishing input – output relationships for unlabelled raw sensor data. Unsupervised ML algorithms, such as K-means, cluster data into separate classes based on its characteristics. “K-means is fairly well established in PD studies that seek to identify subtypes of PD, such as those patients who are tremor dominant versus those with rapid motor function decline and cognitive impairment.” [20]

Of the 17 wireless sensor PD studies conducted during 2001-2016 and reviewed by Kubota [20], only 2 used unsupervised methods in combination with supervised, to classify bradykinesia (slowness of movement) and analyse the link between PD and mild cognitive impairment. Between 2018-2021, 7 deep learning (DL) studies have been conducted for PD; two of these used a combination of supervised and deep learning approaches.

Deep learning learns representations of data with multiple levels of abstractions and is used for handling data without labels. Two different types of deep learning architectures have been explored for time – series sensor data in general and PD wearable data in particular. The bulk of research focussing on Convolutional Neural Network (CNN), which is now considered a state-of-the-art approach to modelling human activity, and on Long Short-Term Memory (LSTM), an architecture of artificial recurrent neural network (RNN), which is considered the best architecture for sequential data, such as time series.

The Convolutional Neural Networks (CNN) have been combined with the application of autoencoders (unsupervised learning techniques and data compression mechanisms that learn to map input data to output data automatically instead of being engineered by a human). For example, autoencoders have been used for denoising the sensor signal by Mohammadian et al., 2018 [21].

Sigcha (2020) [7] summarised that “recent studies propose the use of DL models for HAR (human activity recognition) and FoG (freezing of gait) detection. When working with sensor signals, the authors have successfully used deep networks with CNN and fully connected neural networks, while CNN work as an automatic feature extractor, the fully connected layers are used for classification.”

In the application of DL to detect Freezing of Gait events and several spatio- temporal gait parameters, such as stride length, DL models have performed at a better sensitivity and AUC (area under the curve – an indicator of the model performance) levels than supervised learning. San-Segundo et al., 2019 [22], has achieved AUC of 93.1% for FoG event detection based on the Daphnet dataset, compared to the AUC of 89% achieved by Mancini et al, 2021 [8] which used threshold based algorithm.

In 2018, Zhao, 2018 [13] used the Long Short-Term memory (LSTM, architecture that extends memory of neural networks), to rate the severity of PD disease from gait information using the sequential data of Vertical Ground Reaction Force (VGRF) recorded by foot sensors. The study developed a two-channel model combining LSTM and CNN to learn the spatio-temporal patterns behind the gait data. More recently, Hssayeni et al, 2021 [3] used deep learning LSTM model to estimate the severity of dyskinesia in PD patients on a dataset of 14 PD patients. The study addressed this as a regression problem, rather than a classification problem. The study achieved a high correlation (r= 0.86) with the scores assigned by a neurologist.

Despite higher accuracy, Camps et al [12] pointed out that deep learning models are difficult to train, with respect to time and computation power, especially in the real-time implementation scenarios. One of the suggestions to address this was through “generating adjustable models” that would allow a user a trade-off between performance against an overhead of retraining a model.

Mancini et al [8] summarised the challenge with deep learning as follows: “despite the higher sensitivity in detecting the occurrence of even shorter FOG episodes compared to the previous method (an accuracy above 90% was achieved), the deep learning approaches may require a higher computational cost, requiring up to several seconds from the occurrence of the episode to its detection, making those algorithms not suitable for real-time interventions, such as cueing [auditory signal that aims to interrupt a freezing of gait episode].” However, these constraints will not remain in place for long, as new types of floating – point powerful microcontrollers could allow for the use of ML models in real-time and make them suitable for wearables.

# Theory / Methodology

The aim of the study is to evaluate whether deep learning models can be used for wearable PD devices and achieve better levels of accuracy than state of the art supervised machine learning models.

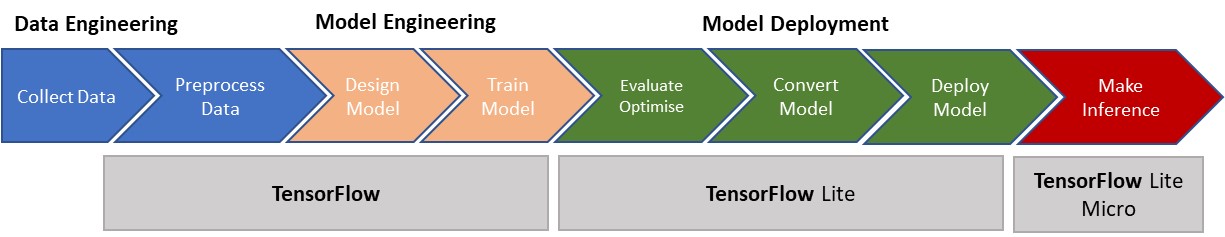
The size of the deep neural nets is typically measured in MB and therefore too large to fit onto constrained wearable devices. This study uses the TinyML framework to optimise and compress the models. The ML framework uses quantization and pruning methods to convert pre-trained models from TensorFlow to TensorFlow Lite, and further into TensorFlow Micro. This study applies the conversions and evaluates the size and accuracy of the model.

The objectives of the study:

* Perform supervised ML models and deep neural networks models for PD use cases, such as monitoring of symptoms (dyskinesia was chosen for the purpose of this study) and classification of activity.
* Compare accuracy of the models, including sensitivity and specificity.
* Compress and optimise neural nets using TinyML framework, to minimise storage and memory footprint.
* Perform inference on a test set, and compare the accuracy of the compressed models, aiming to minimise the loss of model accuracy.
* Discuss results with a PD clinician to understand the acceptance of the proposed model performance.

## Research Methodology and Implementation

This study followed the Machine Learning workflow outlined in the Harvard TinyML Applications course, focussing on data engineering, model engineering and model deployment, as described in **Fig. 3**. Arduino Nano 33 BLE Sense microcontroller with Tiny Machine Learning Shield was used as an edge device to load the model.



**Figure 3 Machine Learning Workflow**

Source: Applications of Tiny ML [23]

**Data Engineering Considerations**

The MJFF (Michael J Fox Foundation) Levodopa Wearable Sensors Dataset [24] was used for this study. The data described in **Fig.4** was collected during 4 days in 2015, using 5 Shimmer3 sensors that contained accelerometer on wrists, ankles and back. The data was collected from 17 patients, of which were 5 female (aged 54-80) and 12 male (aged 55-77), although data from one of the patients proved to be unusable due to a large number of unavailable values (NaNs), leading to the final total of 16 patients. The dataset includes a total of 4 days of data, of which 2 days of labelled data collected in the controlled Lab environment and 2 days of data collected in the home environment.

Text

Description automatically generated with low confidence

**Figure 4 MJFF Levodopa Wearable Sensors Dataset (2015)**

Source: Synapse, MJFF repository [24]

The time series sensor dataset was labelled for several phenotypes including tremors, dyskinesia (upper & lower limbs) and bradykinesia (upper limbs only). Clinicians classified the data according to the severity of the symptom, using the scale of 0-4. The study focussed on one symptom – dyskinesia – aiming to classify the full scale, using the labels that were produced in the lab environment.

There are several limitations and potential biases inherent in this dataset:

* Limited number of PD patients was included in this MJFF study. This means that the findings reached during this study will not be conclusive and will need to be verified on a larger dataset.
* Small number of patients made prediction of some use cases, for example, forecast of UPDRS scores based on the sensor data, impossible, as only a small number of cases will be available for each classification group.
* The patient dataset has only 4 female PD patients, skewing the results toward PD male patients.
* Only data from accelerometers was collected for the Levodopa Sensor dataset. This limits the application of the results of this study, as increasingly IMUs with at least 6-axis, including accelerometer and gyroscope sensors, are used for PD wearables studies.

Nevertheless, MJFF Levodopa Wearable Sensors dataset is one of the few labelled sets available for PD researchers, and it allows to test classification and regression models for several different types of symptoms, beyond Freezing of Gait that has been the focus of most deep learning studies in recent years [7], [12], [25].

Despite its limitations, the small size of this dataset also allows to test the application of deep learning models to medical data, which is subject to the challenge of class imbalance, where only a small volume of data is available for each symptom, compared to large volume of data that is “symptom free”.

**Model Engineering Considerations**

The study set out to compare accuracy and performance for supervised machine learning and deep learning models.

Supervised ML Random Forest ensemble was selected for the classification task. This is one of the most widely used and versatile supervised ML algorithms used for both classification and regression, and tested for PD classification in the studies of Ricciardi et al, 2020 [26], Zhang et al [27]. While it is not always the best performing algorithm compared to other tree-based algorithms, such as Gradient Boosted Trees (GBT) and Ada-Boosting (ADA-B) [26], it is one of the easiest algorithms to use in terms of data preparation and ease of interpreting the results.

For deep learning model engineering, this study selected two types of Deep Neural Networks – convolutional neural network (CNN) and Recurrent Neural Networks (RNN) – Long Short-Term Memory (LSTM) that is commonly used for time series. Both have been extensively used in the recent PD research by Mohammadian et al [21], San Segundo et al [22], and Sigcha et al [7].

Convolutional Neural Networks have been successfully deployed for human activity recognition using the following capabilities of different layers:

* 2D convolutional layers are used for identifying features and patterns. Heim et al (2021) [28]
* Dense layers are used for backpropagation and calculation of weights.
* Dropout layers are used for regularisation and avoiding model overfitting.

Heim et al [28] concluded that the best optimisation is achieved during the design process, choosing the best combination of layer types and dimensions. For example, it was concluded that latency optimisation is optimised through the choice of input connections. Faster latency is achieved if a number of inputs is an even number, or divisible by 4.

Recurrent Neural Network with Long Short-Term Memory (LSTM) was used for time series data as this type of deep learning models can track temporal data patterns.

**Model Evaluation Considerations**

The MJFF dataset is characterised by the presence of unbalanced classes, a common feature of medical AI datasets. In the models with unbalanced classes, accuracy is no longer a good measure of performance. Prediction of all classes that do not have a feature, e.g., a dyskinesia symptom, will still yield high accuracy, as was demonstrated and described later in this study.

There are several potential solutions to this problem:

* Collect more data for underrepresented classes - this was not an option for our study in the absence of a live trial option.
* Create copies of training samples for the underrepresented classes. For the dataset where symptoms represent as little as 5% of the total, this would increase the overall number of training sample.
* Transfer learning and data augmentation techniques. For example, augmented copies of small classes can be created using such functions as SMOTE (Synthetic Minority Over-sampling Technique) data augmentation algorithm, developed by Chawla et. al in 2002 [29]. SMOTE addresses class imbalance by artificially generating new examples of the minority class using the nearest neighbours of this class. No data augmentation was applied in this study.
* Finally, the model can be trained for specificity (proportion of true negatives in the total number of negatives) and sensitivity (proportion of true positives in the total number of positives). A model that always predicts a symptom will have a sensitivity of 1 and a specificity of 0. The ideal outcome is where both are equal to 1, however, in practice, medical AI models are trained to achieve a trade-off. Setting the trade-off at 0.5 means that both are equally important.

The training for specificity and sensitivity was the technique selected for this study.

**Model Deployment Considerations**

Once developed and validated, the model for embedded devices and microcontrollers needs to be optimised for performance, power consumption and latency.

TinyML has emerged as an approach to optimising machine learning models for constrained devices, that combines improvements in hardware, software and algorithms to perform analytics and inference on the battery operated edge devices [30].

TensorFlow Lite (TF Lite) and TF Lite Micro have been established as the main TinyML frameworks supported by multiple large developer communities globally. These are built within TensorFlow, which is one of the most widely used deep learning frameworks, with Python front end and C++ code in the core. Consequently TF Lite and TF Lite Micro were therefore chosen for this study, although other approaches such as Edge Impulse and Neuton have recently been developed to give smaller size, lower utilisation of Flash, RAM and inference time [31]. It is also possible to convert models developed other frameworks such as PyTorch and ONNX (Open Neural Network Exchange).

In our study, this was done by using conversion tools post-training: a TensorFlow Converter compressed a model developed in TensorFlow to the format suitable for running on an edge device.

* TensorFlow Lite is a runtime optimised for Android, iOS and embedded systems that run on a variant of Linux, for example, Raspberry Pi. During the conversion, many of the memory consuming elements of the neural network are removed, without compromising accuracy. Quantization can take place during this conversion, where data is reformatted from high-memory consuming float32 into a lower-memory format int8.
* TensorFlow Lite Micro is a core enabling technology for TinyML framework; it is built on top of TensorFlow Lite to downsize the model even further to work on microcontrollers.

Three main methods to compress the models have demonstrated inference efficiency, including quantization, weight sharing and network pruning.

A study by Han et al (2016) [32] implemented a three stage “deep compression” pipeline, combining pruning, quantization and Huffman coding. This approach targeted latency – sensitive applications running on mobiles, which required real – time inferences. The study concluded that pruning reduced the number of connections by 9x to 13x; Quantization then reduced the number of bits that represent each connection from 32 to 5, reaching compression rate between 27x and 31X. Finally, application of Huffman coding reduced overall storage requirements by 35X to 49X without reduction in accuracy. Combined, pruning and quantization compressed the network to 3% of the original size.

Heim et al (2021) [28] have conducted research that optimised neural nets with a focus on “perceptible metrics” – the metrics that the end-user is exposed to, such as inference latency and energy consumption. Their study experimented with 8-bit quantization, chosen so because it is supported by most MCU (compared with 4-bit or adaptive rate). It applied quantization and specialised kernel to two models - LeNet and ResNet - and concluded that operating on a floating point and fixed point made significant difference, achieving memory footprint reduction of 73% while losing only 0.05% of the accuracy when tested on the host (PC that was used to design and optimise the model). With regards to the latency, while the floating-point unit (FPU) was disabled, the unoptimized model achieved a 4x faster inference; applying software acceleration with the specialized neural network kernel (CMSIS-NN) to the quantized (optimised) model achieved additional 4x improvement in latency.

The following limitations and biases have been identified in our study:

* While optimised models could be deployed on Arduino Nano BLE with Machine Learning to test, it was not possible to test the accuracy of inference without access to the test cohort of PD patients.
* The MJFF dataset was based on the readings of 5 sensors worn on each limb and on the lower back. Real-life deployments would aim to minimise the number of sensors worn on a patient, and if a limited number of sensors is worn, the ML models intended for use on a real wearable will need to be retrained for the right placement of sensors.

## Expected Outcome and Impact

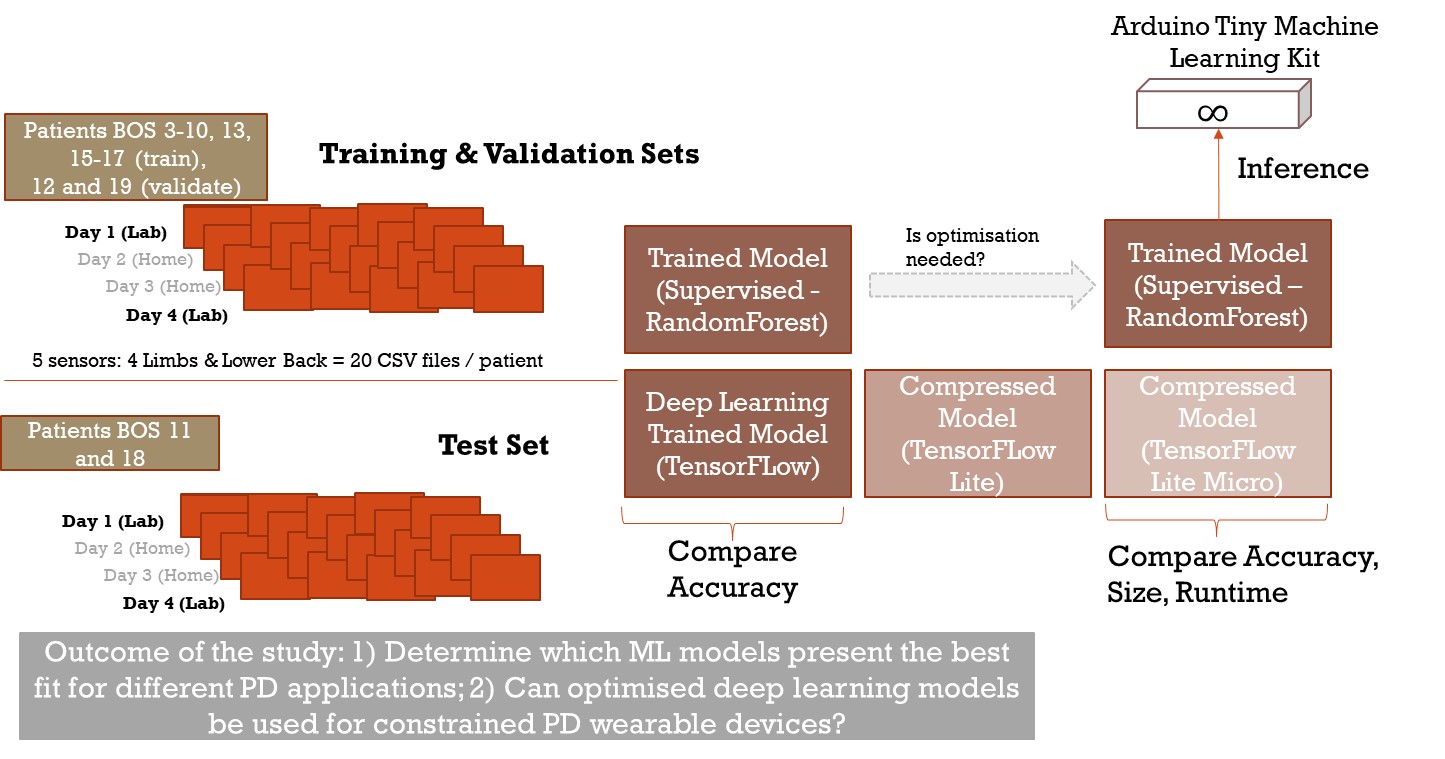
The desired impact of this research is to improve the feasibility of using wearables for monitoring Parkinson’s Disease, and aid with the faster adoption of wearable systems for PD applications.

This research aimed to strengthen the understanding of machine learning model fit for wearables used for Parkinson’s Disease applications. Use of PD wearables in the home monitoring environment requires accurate and robust predictions, as well as ability to deploy the chosen model onto a constrained wearable device. The aim of the study was to demonstrate that this could be achieved via the use of deep learning models.

The expected outcome of this research was to evaluate the best optimisation methods for supervised and deep learning models, aiming at the smallest memory footprint and compute requirements, using TinyML framework.

# ML Workflow Implementation and Optimisation Discussion

The details of the machine learning workflow used in this study are presented in **Fig.5**.



**Figure 5 ML For PD Study: Modelling Approach**

## 4.1. Data Engineering Discussion

Training medical classification algorithms is more challenging than a typical image classification algorithm, which is based on hundreds of thousands of images. The following key challenges, which apply to medical data engineering, have been addressed in this study :

* Multi-task and Class imbalance challenge. For dyskinesia, this study set multi-class labels to model 4 levels of severity (0-4) captured and labelled during the 2 days of the lab-based study. Even for patients with high severity of PD, however, such symptoms as dyskinesia do not occur all the time. In the MJFF study, for example, dyskinesia symptoms were observed during such activities as folding sheets but not during walking. Thus, there were a lot more examples of asymptomatic signal than sensor signal with dyskinesia.

The weighted loss/resampling for class-imbalance was applied to address this challenge, as part of the model.fit class\_weight built-in method. This method provides a weight or bias for each output class, based on the proportion of each class’s instance in the total number of label instances:

*n\_total\_label\_samples / n\_class\_label\_samples*.

A dictionary was created as follows, demonstrated the scarcity of some classes, especially class 4, that was present at a ratio of 7,780 : 1 compared to class zero.

class\_weight = {0: 1., 1: 48., 2: 137., 3: 543., 4: 7780.}

In our study, only data from Boston cohort who wore Shimmer sensors on all limbs was included in the modelling, to ensure the consistency of sensor measurements. Predictions for dyskinesia were modelled using lower limb sensor data only. The NY cohort data was excluded as the study focussed on lower limb gait analysis, while the NY cohort patients collected sensor data from smartwatches and smartphones, worn on both hands.

Each day of the sensor dataset has several million readings per patient. Each patient’s datasets for days 1 and 4 were stacked and concatenated to create a training dataset. These are the 2 days where symptoms were labelled and validated in a lab environment. Although it would be interesting to test the model on the data collected in the home environment, the absence of labels would make it impossible to validate the results.

Each data slice was combined into a sequence length of 128-time values – at 50Hz, this translates into 2.5 seconds of data. Each time value was associated with a particular label. In order for each time slice to have the right label, the statistical mode was calculated for the labels, picking the label with the most occurrence in each sequence of sensor readings.

Finally, the data was normalized, or rescaled from the original range to use the values within the range of 0 and 1, as follows:

*X\_train[training\_set\_columns] = x – np.min(x) / (np.max(x) – np.min(x))*

The normalization is typically applied to time series data with variable scales. While the scale is not drastically different for X, Y and Z-axis of accelerometer readings, the conversion facilitates the training on different patient datasets.

## 4.2. Model Engineering Discussion

This section describes the choice of model, model architecture, choice of parameters, data split approach, and discusses initial results and performance.

## 4.2.1. Convolutional Deep Learning Model Architecture and Parameters

The 10-layer Sequential Convolutional 2-dimensional (2D) Neural Network Architecture was chosen for the study (see **Fig. 6**). 1-dimensional (1D) CNN models can work equally well for simpler time-series data.

The model was developed using the TensorFlow version 2.5.0; Jupyter Notebooks with Python version 3.7.4. The following libraries were used for the model: Pandas, Numpy, Tensorflow, and Keras.

Chart, bar chart

Description automatically generated

**Figure 6 CNN Model Architecture Summary**

The following sequence of layers was chosen for training the PD sensor time series data:

* In a sequential model, the output of each layer is passed directly to the next layer.

The first layer is **Conv2D convolutional layer**. Its role is to take the raw accelerometer data directly and extract the basic features, which are then interpreted by subsequent layers. Each measurement has three values that represent the X, Y, and Z axes. The Conv2D layer then slides a window across the data to determine whether a given feature is present in that window.

* The arguments to the Conv2D() function determine how many features are extracted by the layers:
  + The first argument establishes how many filters the layer has. Our model uses 8 filters, each learning to identify a particular feature in the raw data and creating an output as a feature map that shows where each feature occurs in the input.
  + The second argument to Conv2D() provides the dimensions of the sliding window, set to (4, 3) in this study. This means that the features span four consecutive accelerometer measurements and all three axes, and the layer generates features that show a change in acceleration over time.
  + The input\_shape argument sets the input’s shape. In our model, it was set to (seq\_length, 3, 1), where seq\_length is the total number of accelerometer measurements that are passed in. It was set to the default value of 128 – divisible by 4, as described earlier. The data was reshaped from the original tabular to be fed into the model in 4D format.

This layer outputs the shape (batch\_size, 128, 3, 8). The value of each of its 8 feature channels shows the degree to which a feature was present in that location of the input.

* + The activation function for each node is set as “relu” (Rectified Linear Unit), where a gradient is set to zero for all inputs less than zero, and a positive input is output directly. The use of this function is now well established for deep learning models, addressing the problem of “vanishing gradients” for large networks (where backpropagated error decreases very dramatically and prevents deep layers from learning effectively), while retaining most benefits of linear functions.
* The **MaxPool2D layer** then transforms the big input sensor into a smaller output by squeezing the output of Conv2D layer into a high-level representation of the most relevant features it contains. A combination of Conv2D and MaxPool2D merges 3 accelerometer axes into a single value.
* **The Dropout layer** uses a regularization technique to avoid the model overfitting the data, by setting some of a tensor’s values to zero during training. In our case, Dropout(0.1) sets 10% of the values to zero, randomly removing these values. Another outcome of this technique is to make the neural network more robust, teaching it to deal with unexpected noise and variation.

The model in this study set 2 Conv2D and MaxPooling layers, as the time series input is quite simple, and to keep the size of the model small. For more complex model, additional convolutional and pooling layers can be added.

* The **Flatten layer** transforms a multidimensional tensor into a tensor with a single dimension. In this case, our (14, 1, 16) tensor is transformed into a single dimension with shape (224).
* This single tensor feeds into a **Dense (also known as Fully Connected) layer** with 16 neurons, which considers all data at once and learns the meanings of various combinations of inputs. The output of the Dense layer is a set of 16 values that represent the original input in a highly compressed form.
* The final layers decrease these 16 values into 5 classes, corresponding to values 0-4 of the PD symptom severity. Another dropout layer is added first, followed by **a final Dense layer** with five neurons representing each class of the symptom. Each neuron is connected to all 16 of the outputs from the previous layer. During training, each neuron learned the combination of activations from the previous layer. This layer was using a "softmax" activation function, which means that the output is a set of probabilities that add up to 1. The final output is the model’s output tensor.

**Model: "Sequential\_8"**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_15 (Conv2D) (None, 128, 3, 8) 104

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_15 (MaxPooling) (None, 42, 1, 8) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_23 (Dropout) (None, 42, 1, 8) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_16 (Conv2D) (None, 42, 1, 16) 528

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_16 (MaxPooling) (None, 14, 1, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_24 (Dropout) (None, 14, 1, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_8 (Flatten) (None, 224) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_16 (Dense) (None, 16) 3600

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_25 (Dropout) (None, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_17 (Dense) (None, 5) 85

=================================================================

**Figure 7 CNN Model Architecture Summary**

## 4.2.2. Long Short-Term Memory Recurrent Neural Network Model Architecture and Parameters

Another deep learning model used in this study is LSTM – Long Short-Term Memory model, which is a variant of recurrent neural networks (RNN). RNN models are composed of multiple channels, where the information from the previous steps is retained and passed to the future steps. The long-term path takes inputs from the previous long-term memory outputs and is also updated from inputs from short-term memory path. Thus, RNN models are frequently used for time-series data.

Similar to the CNN model described above, LSTM model uses a sequence of layers. The following layers were chosen for training our PD sensor time series data:

* The first layer is a **LSTM layer**, which takes the raw accelerometer data directly and extract the basic features.
* This layer feeds into a **Dense layer** with 32 neurons, which considers all data at once and learns the meanings of various combinations of inputs. The output of the Dense layer is a set of 16 values that represent the original input in a highly compressed form.
* The final **Dense** layers decrease these 16 values into 5 classes, corresponding to values 0-4 of the PD symptom severity. The final output is the model’s output tensor.

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm (LSTM) (None, 50) 10800

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 32) 1632

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 5) 165

=================================================================

**Figure 8 LSTM Model Architecture Summary**

## 4.2.3. Data Split into Training – Validation – Testing sets

The dataset in this study was split into training, validation and test sets. The training set was split into training and validation: training set used for developing the model, while the validation used for model optimisation, e.g., hyperparameter training. The test set was used to test the model.

There are several challenges in splitting medical datasets [33] that needed to be addressed, the main one being patient overlap. In the traditional split of data into training, validation and testing, the data from the same patient is split randomly, and can feature in all 3 sets. However, this can lead to the loss of data independence, if a deep learning model memorises some rare or unique aspects of the data and associates the label with this aspect instead of the feature of the outcome. This would lead to the overly optimistic performance of the model. The recommended solution adopted in this study was to include each patient’s data only in one data set, without any data overlap – e.g., sensor data from 12 patients was be used for model training (75% of the total), data from 2 patients for validation, and data from 2 patients for testing (12.5% of the total each).

In the MJFF dataset the sensor data for each patient is contained in a separate data file. It was merged into target training, validation, and test sets.

While random selection of data is preferable, it was important to ensure that training dataset included data from patients with different degree of PD severity. For example, three patients in the MJFF cohort had H&Y score of 3 (patients 3\_BOS, 11\_BOS and 17\_BOS) and 2 had the score of 4 (patients 18\_BOS and 19-BOS), with the rest scored 2 on H&Y scale. Data from patient 14 was excluded from the set, as it contained largely NaN values and was deemed not useful for training. The training, validation and testing data was pre-selected to include examples from both genders and from different degrees of severity, to provide a variety of symptoms for each dataset (see **Fig. 9**):

Training (75%): Patients 3\_BOS – 10\_BOS, 13\_BOS, 15\_BOS– 17\_BOS

Validation (12.5%): Patients 12\_BOS and 19\_BOS

Test Dataset (12.5% of all patients): Patients 11\_BOS and 18\_BOS

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| Gender | F | F | M | M | M | M | M | M | M | F | M | M | F | M | F | M | M |
| PD H&Y scale | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 4 | 4 |
| Model dataset | Tr | Tr | Tr | Tr | Tr | Tr | Tr | Tr | Te | V | Tr | n/a | Tr | Tr | Tr | Te | V |

**Tr** = Training dataset; **V** = Validation dataset; **Te** = Test dataset

**Table 1 Modelling Data Split for MJFF BOS Patient Data sets**

## 4.2.4. Supervised Learning Model

To compare different types of models, this study used Random Forest supervised learning classifier. Random Forest is an ensemble of decision tree algorithms, used on tabular data. The MJFF dataset converted into Pandas dataframes fit this format requirement.

The Random Forest algorithm takes bootstrap samples from the training dataset, sampling with replacement for a large number of decision trees. Each decision tree is fit on a slightly different dataset and is left unpruned. Predictions are averaged across all trees, where a prediction for each class label is based on the majority vote across the trees in the ensemble. This leads to a better overall performance than predicting for a single decision tree.

The evaluation of the model used k-fold cross-validation, Random Forest classifier achieved training accuracy of 0.973 for a single patient set, and testing accuracy of 96.97% - only slightly worse than neural networks described in the following sections. However, similar to the deep learning models, it suffered from class imbalance problem, as described in the Results section.

## 4.2.5. Machine Learning Model Performance

Convolutional Neural Network model was compiled using the following key parameters:

* optimizer = tf.keras.optimizers.Adam (Adaptive Moment Estimation). Adam is a stochastic gradient descent method that uses momentum (less oscillation while converging to the local minima) and adaptive learning rate (e.g., starting with bigger steps and finishing with smaller steps) to converge faster. Adam is based on adaptive estimation of first-order and second-order moments, and currently performs in the best and most consistent manner [34].
* loss = 'sparse\_categorical\_crossentropy'. The error between the predicted output probability and the actual output probability is measured by computing a loss function. The sparse categorical cross entropy function applies to multi-class models.

The initial CNN model was prototyped and trained on the data from one patient (BOS\_10). The following results were achieved: training set accuracy: 0.9936, validation set accuracy: 0.965, and test set accuracy of 0.9698, which means that correct class was predicted 96% of the time. Training the 2D - CNN model on a full 12-patient dataset for dyskinesia, using left lower limb accelerometer data, yielded the lower training set accuracy of 0.9720, validation set accuracy of 0.9868, and test set accuracy of 0.97, where correct classes were predicted 97% of the time.

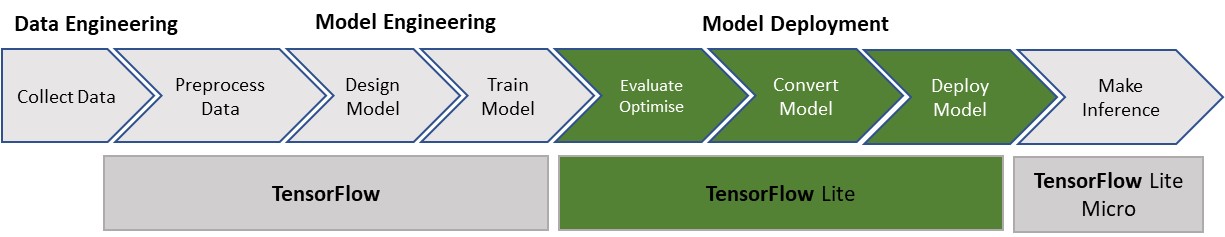
The confusion matrix () function output the tensor, where columns represent predicted labels, and the rows are actual labels. Without class\_weights method, the model reached 97% accuracy, however, it was accurate only for the class “0” label, which made it unusable for predicting the actual symptoms of dyskinesia. Full set of accuracy results is presented in **Fig. 10**.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Training – Validation – Testing Accuracy** | **TF Model Size** | **TF Lite / TF Lite with Optimisation** |
| **CNN 2D – trained on 12 patients** | 97.20% - 98.68% - 97% | 102 KB | 20.2 KB /  9.7 KB |
| **LSTM trained on 12 patients** | 97.54% - 97.52% - 96.98% | 368 KB | 87.7 KB/  53.8 KB / |

**Table 2 Deep Learning Models: Accuracy and Size**

## 4.3. Model Deployment Discussion

The next step in this study was to optimise and convert the deep learning model and prepare it for deployment to an edge microcontroller device (see **Fig.11**).



**Figure 9 Machine Learning Model Flow: Model Compression Step**

## 4.3.1. Optimisation and Compression: TensorFlow Lite

The main objective of the model optimisation and compression is to allow its deployment onto wearable mobile devices and wearable sensors that use microcontrollers. This is achieved by converting the model from its fully trained size in TensorFlow, using compression and quantization techniques of the TensorFlow Lite, Google’s open-source ML framework, designed to run in desktop environments and cloud servers.

Unlike cloud servers, deployment of ML models on wearable devices requires a significantly smaller size of models, and to respond to this need, Google has developed the new library TensorFlow Lite in 2017. It was further optimised in TensorFlow Lite Micro (TFLM), developed in 2020. The comparison between the frameworks is presented in **Fig.12**.

|  |  |  |  |
| --- | --- | --- | --- |
|  | TensorFlow | TensorFlow Lite | TensorFlow Lite Micro (TFLM) |
| Training | Yes | No | No |
| Inference | Yes (not efficient on the edge) | Yes | Yes |
| Number of ops | ~1400 | ~130 | ~50 |
| Base Binary Size | 3MB+ | 100 KB | ~10 KB |
| Base Memory Footprint | ~5MB | 300KB | 20 KB |
| Native Quantization | No | Yes | Yes |
| Needs an OS | Yes | Yes | No |
| Memory mapping of models | No | Yes | Yes |

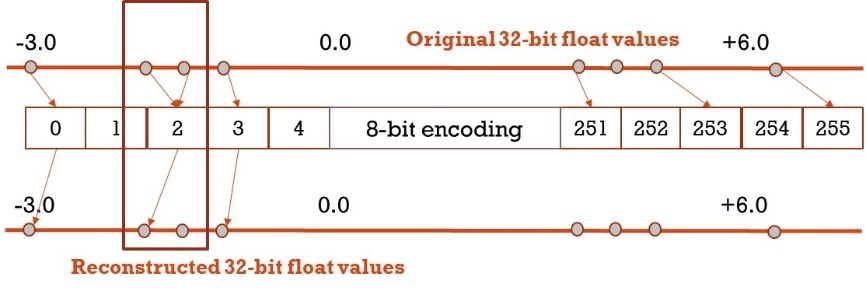
**Table 3 TensorFlow Frameworks: Comparison of TF, TFLite and TFLite Micro**

Source: Harvard University Online “Deploying TinyML” course [35]

Each TensorFlow model is represented as a computational graph – an abstract directional graph that represents computations and specifies relationships between constants, variables and operations. Quantization covers a number of different techniques that store these numbers and perform calculations on them using more compact formats than 32-bit floating point numbers [35]. For wearables, conversion in 8-bit integers has been applied.

The floating-point numbers are used to train the model through the use of stochastic gradient descent, as these allow for the high level of precision. Using more compact formats is possible due to the deep neural networks’ ability to deal with higher levels of noise – the lower precision numbers are treated as a source of noise, while still producing accurate results.

Quantization takes advantage of the fact that model parameters, e.g., weights, fall into a small range of normally distributed numbers, e.g., from -3.0 to 6.0. The file size is decreased by means of storing the min and max for each layer and then compressing each float value to an eight-bit integer representing the closest real number in a linear set of 256 within the range (**see Fig. 13**). In this example, a 0 byte would represent -3.0, a 255 represents 6.0, and 128 would represent ~1.5. The resulting size of the model file will decrease by 75%.

****

**Figure 10 TensorFlow Model Quantization: Compressing Floating Point Numbers [36]**

Once the model is loaded onto the edge device, the weights are converted back to floating point – assuming a microcontroller will be supporting floating point operations - and the previous model code can be applied without any modifications. If two floating points are so close together that they are encoded into the same 8-bit bucket, some resolution will be lost, and a small amount of accuracy be lost. This is the trade-off in using quantization techniques.

The process of converting the TensorFlow model using APIs was as follows:

1. Save the full model as Keras .pb model, Google ProtoBuffers format that includes both metadata (or structure of the computational graph), and weights and biases of the model.
2. Convert the model using TFLiteConverter.from\_keras\_model. The model is stored in FlatBuffer TensorFlow Lite file format (.tflite file).

To compare the size of the supervised model, the Random Forest model is saved using the pickle format, Joblib library. No further compression has been applied to the Random Forest model.

## 4.3.2. Further Model Compression: TensorFlow Lite Micro

Despite significant compression achieved through the use of TensorFlow Lite, further compression was needed for deploying neural network models into microcontrollers.

Our TF Lite CNN model described in the previous section was 102KB, further compressed to 20KB. However, in addition to the model itself, the microcontroller also requires allocation of memory for the application code, libraries, working memory and input data buffer. The example of memory usage is presented in **Fig. 14.** Unlike TensorFlow ProtoBuffer model, the TensorFlow Lite and TensorFlow Lite Micro models do not require dynamic memory to be allocated and copied at runtime.

TensorFlow Lite’s standard library, for example, is ~400MB in size, and can take as much as 1GB of RAM to run. The TF Lite Micro addresses this challenge by stripping all but the most essential elements for running ML on microcontrollers in general, and the Arduino Nano BLE board, used for this study, which has only 1MB of flash memory and 256KB of RAM.

**Chart, bar chart

Description automatically generated**

**Figure 11 Example of Microcontroller Memory Usage [35]**

The TFLite Micro binary is about 1MB in size and the core runtime of its library takes 16KB. This simplification comes at a price – troubleshooting TF Lite Micro is complicated as all functionality for plotting and debugging is removed. Thus, TF Lite models are trained on devices with more computational resources and are then ported to the microcontroller. Only inference is performed on the microcontrollers themselves.

Non-volatile Flash memory is where the weights and the model code (e.g. in C or C++) are stored. These are over-written only during re-programming, never during the programme execution. The size of the model weights and code is therefore limited by the size of the microcontroller Flash memory (minus 2KB bootloader).

RAM stored temporary variables, e.g., input and output buffers, and intermediate tensors. This is an important limitation when deep learning models are concerned. If time series data is pre-processed, for example, being converted into a spectrogram and classified as an image prior to being fed into a 2D CNN model, it will affect the usage of the RAM. Variance in the length of a window is equally important, with bigger chunks of data need higher input buffers and consequently usage of RAM.

Last but not least, many microcontrollers also do not have an OS, running on bare metal instead. This means they cannot rely on ***malloc*** or ***new*** standard C/C++ memory APIs that embedded systems typically rely on.

To solve the memory challenge, TF Lite Micro allocates a chunk of contiguous memory to the Interpreter, to avoid the fragmentation during the life-cycle of the device, which for some healthcare wearables could mean months and even years of operating on a battery.

Furthermore, TF Lite Micro strips down the operators that are not needed for the model – as many as 1400 operators are supported in TensorFlow. Our 2D CNN and LSTM models, for example, only use 5 different types of layers – Conv2D, MaxPool 2D, Dense/ Fully Connected, Flattening and Softmax. An op resolver is added for each of these, bringing down the number of operations to under 200, and reducing memory footprint by as much as 30% [36].

For the implementation, following the conversion of the model into TFLite format, the model is converted into a hexdump file, which is a binary file with its contents represented by hexadecimal values. Finally, this hexdump file was flashed into the flash memory of the microcontroller. The microcontroller is consequently ready to perform inferences on the embedded device.

## 4.4. Results Discussion

**Model Performance: Accuracy**

The first objective of the study was to evaluate the performance of deep learning vs supervised learning model. This was evaluated using the accuracy metrics from the TensorFlow Keras Metrics library tf.keras.metrics. Most studies for ML for PD report a range of metrics, including overall accuracy, AUC, sensitivity and specificity.

The models were evaluated against the standard metrics used in the data science community, where accuracy, precision, recall and F1 score are commonly used.

* *Accuracy(A) = (tp + tn) / (tp + tn + fp + fn)*, where tp = true positives; tn = true negatives, fp = false positives, and fn = false negatives
* *Precision (P, or Positive Predictive Value) = tp /(tp + fp)* – of all examples predicted positive, how many are positive
* *Recall ( R ) = tp/(tp + fn)* out of all positive examples, how many are predicted positive
* *F1 score: (2 \* R \* R)/ (P+R)* a harmonic mean of precision and recall that helps to balance the two scores. A very small precision or recall score will result in lower F1 score.

The overall accuracy of all three models in this study was comparable, at 97%, but CNN performed best on all other metrics. AUC was 99% for the CNN model, 96% for Random Forest model, and 84% for LSTM model with class weights applied (see **Fig. 15**).

In the models with unbalanced classes, however, accuracy is no longer a good measure of performance. Poor prediction of the classes with a symptom, e.g., dyskinesia, will still yield high accuracy due to the dominance of the Class 0 (no symptom), which accounts for 97% of all labels in the MJFF dataset.

Moreover, in medical AI, the common measures of model performance are sensitivity (proportion of true positives in the total number of positives) and specificity (proportion of true negatives in the total number of negatives).

* *Sensitivity = tp / (tp + fn)* - out of all people **with** disease, how many received positive results
* *Specificity = tn /(tn + fp)* – out of all people **without** disease, how many received negative results

A model that always predicts a symptom will have a sensitivity of 1 and a specificity of 0. The ideal outcome is where both are equal to 1, however, in practice, medical AI models are trained to achieve a trade-off. Setting the trade-off at 0.5 means that both are equally important. For a person that has a disease or symptom, Recall and Sensitivity will be the same, but Precision and Specificity will be different.

In case of PD, it will need to be agreed with a PD specialist as to where the trade-off between sensitivity and specificity should lie. If a patient experiences symptoms but these are not captured (lower sensitivity), then progression of PD and impact of medicine will be underestimated. If patients’ symptoms are over-predicted (lower specificity), the progression of PD will be overestimated.

Consequently, the two deep learning models were re-trained for specificity and sensitivity, with the results presented in **Fig. 15**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Categorical Accuracy | Precision | Recall Score | F1 Score | Specificity | Sensitivity | AUC |
| LSTM | Train: 69.41%  Val: 56.52% | 73.79%  56.51% | 66.94%  55.73% | 63.68%  49.65% | 89.3% | 55.73% | 94.38%  84.04% |
| CNN | Train: 97.54%  Test: 98.68% | 97.54%  98.68% | 97.54%  98.68% | 97.5%  98.7% | 99.3% | 97% | 99.82%  99.34% |
| Random Forest | Train:  97.6%  Test:  96.86% | 96.98% | 96.98% | 96.98% | 96.98% | 96.98% | 96.98% |

**Table 4 Machine Learning Models: Accuracy Metrics**

Analysis of the LSTM model, where the class-weights have been applied and sensitivity and specificity were generalised, was that the mode under-predicted all dyskinesia symptoms for the classes with labels from 1 to 4. Only one class – Class with the dyskinesia severity of 2 – was predicted to some degree of accuracy. Other classes were not, which means that the model performed with a low level of sensitivity and high level of specificity. Similarly, Random Forest model was accurate for Classes 0 and 1, but the smaller Classes 2,3 and 4 were not forecast accurately.

For a practitioner interested in seeing as much information on the patient symptoms, this model performance will be insufficiently accurate, and deep learning model parameters will need to be further optimised, data enhanced, and retrained.

**Model Performance: Size**

The second objective set by this study was to establish whether deep learning models could be optimised and compressed.

In our study, the CNN model without any optimisations or quantization (model1.tflite) was converted from its original 102KB to a size of 20KB (20,236), a compression of x5 times. RNN – LSTM model without any optimisation was converted to a size of 87.6KB (see **Fig. 16**).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Accuracy (Train- Validate – Test)** | **TF Model Size** | **TF Lite / TF Lite w. Optimisation** |
| **CNN 2D** | 97.20% - 98.68% - 97% | 202 KB | 20.560 KB /  9.7 KB |
| **RNN - LSTM** | 97.54% - 97.52% - 96.98% | 368 KB | 87.84 KB/  53.8 KB / |
| **Random Forest** | 97.3% | 2,016KB | n/a |

**Table 5 Machine Learning Models: Size Metrics**

The model with optimisation using converter.optimizations = [tf.lite.Optimize.DEFAULT] was converted to a size of 9.7KB (9,728), a compression of x10 compared to the original model. This converter quantized model weights. Previously, the default converter was optimising for both size and latency, but the updated default converter optimises for both, while aiming to retain the accuracy.

The LSTM model with optimisation was compressed to 53.8 KB – 5x larger than the 2D CNN model.

This makes CNN model fully suited to the deployment on microcontrollers, while LSTM model may not compress small enough and can only be suitable for mobile OS that can take larger sizes.

Both CNN and LSTM models failed to convert weights to 8-bit precision. The issue has not yet been addressed by the Tensorflow community [[link to Github issue](https://github.com/tensorflow/tensorflow/issues/35194)].

Finally, the newly converted TF Lite models were tested for accuracy compared to the accuracy of the full model. A sample of the data was taken from the Test set and the new TF Lite model was fitted. A different interpreter was used to transform TF Lite model. The generated labels were compared with the actual test set labels. The conclusion was that no loss in accuracy in the compressed model compared with the full TensorFlow model.

With the accuracy of the model contested as a useful indicator, as per discussion above, this metrics would need to be retested once the models are retrained for higher sensitivity.

No inference for dyskinesia was tested on the microcontroller in this study.

# Conclusions and further work

Previous PD research demonstrated that deep learning models could achieve sensitivity rates in excess of 90% compared to 85% rates achieved with the best threshold algorithms. This study set out to test the accuracy of the selected model on the MJFF data set, which was gathered in 2015, to validate these findings.

Using the MJFF dataset presented several challenges. First, only a small number of PD patients – 16 with full sets of sensor data captured - were using Shimmer sensors on their lower limbs in the study. This makes the outcomes of this study too small to be extended to a larger population of PD patients. The dataset is also characterised by class imbalance, where 97% of all labels are Class 0 without symptom.

Nevertheless, MJFF Levodopa Wearable Sensors dataset is one of the few labelled sets available for PD researchers that include several different types of symptoms, beyond Freezing of Gait that has been the focus of most deep learning studies in recent years [7], [12], [25]. Despite its limitations, the small size of this dataset also allows to test the application of deep learning models to medical data, which is subject to the challenge of class imbalance, where only a small volume of data is available for each symptom, compared to large volume of data that is “symptom free”.

Two deep learning models - 2D convolutional neural network and LSTM – as well as a Random Forest supervised algorithm were tested on the data from 12 patients who were tracked over 2 days in the lab environment, focussing on the symptom of Dyskinesia that was labelled in the lab conditions.

A similar overall accuracy was achieved by all three models. AUC was 99% for the CNN model, 96% for Random Forest model, and 84% for LSTM model with class weights applied. However, the models differed widely with regards to sensitivity and specificity. The LSTM model, where the class-weights have been applied and sensitivity and specificity were generalised, under-predicted all dyskinesia symptoms for the classes with labels from 1 to 4. Only one class – Class with the dyskinesia severity of 2 – was predicted to some degree of accuracy. Other classes were not, which means that the model performed with a low level of sensitivity (ratio of true positives) and high level of specificity (ratio of true negatives). For a practitioner interested in seeing as much information on the development of patient symptoms, this model performance will be insufficiently accurate, and deep learning model parameters will need to be further optimised, data enhanced, and retrained.

In addition to the accuracy, deep models have traditionally been too big to fit onto the microcontrollers. The Flash and RAM of the microcontrollers varies, but in most cases the model needs to be under 20KB in size. In this study, deep learning models were compressed using the TensorFlow Lite framework to optimise and quantize the models post-training:

* Significant compression improvements were achieved for both CNN and LSTM models: the size of the convolutional deep learning model achieved during the conversion was to below 10KB.
* This was significantly smaller than the trained Random Forest model, which was at 2MB in its fully trained size.

Conclusion:

* Deep learning models would be a viable alternative to widely spread supervised learning models, if they can be optimised to achieve high levels of sensitivity.
* Despite high levels of overall accuracy achieved in this study, additional mechanisms need to be put in place to improve the performance of deep learning models for small classes.
* Small size of the medical dataset presented a challenge in generalising the findings to a broader set of PD cases. Thus, further work would need to be undertaken to develop larger datasets, and more data would need to be collected for under-represented classes, or synthetic datasets would be generated, in order to make the deep learning models usable for inference on wearable PD devices.

Once the initial technical feasibility of hosting deep learning models, which has been addressed in this study, has been validated, these models will also need to be made more transparent via the use of Interpretable AI, to build up trust among the healthcare professionals.

Following this, deep learning models in combination with Tiny ML hold a big potential for healthcare and Parkinson’s Disease wearables. Deep learning on embedded devices opens the possibility to build highly accurate models for time series data and complement these by other functionality that is now also developed by Tiny ML community. In parallel, some of the new capabilities, such as Keyword Spotting, Visual Wake Words, among others, will open venues for developing new PD applications.

# References

[1] S. L. Kowal, T. M. Dall, R. Chakrabarti, M. V. Storm, and A. Jain, “The current and projected economic burden of Parkinson’s disease in the United States,” *Mov. Disord.*, vol. 28, no. 3, pp. 311–318, 2013, doi: 10.1002/mds.25292.

[2] S. Mair, “Could an Apple smart watch track your Parkinson’s disease symptoms?,” *Parkinson’s Life*, 2021. [Online]. Available: https://parkinsonslife.eu/apple-smart-watch-track-parkinsons-disease-symptoms/.

[3] M. D. Hssayeni, J. Jimenez-Shahed, M. A. Burack, and B. Ghoraani, “Dyskinesia Severity Estimation in Patients with Parkinson’s Disease Using Wearable Sensors and A Deep LSTM Network,” *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, vol. 2020-July, pp. 6001–6004, 2020, doi: 10.1109/EMBC44109.2020.9176847.

[4] M. D. Hssayeni, J. Jimenez-Shahed, M. A. Burack, and B. Ghoraani, “Dyskinesia estimation during activities of daily living using wearable motion sensors and deep recurrent networks,” *Sci. Rep.*, vol. 11, no. 1, pp. 1–12, 2021, doi: 10.1038/s41598-021-86705-1.

[5] G. AlMahadin, A. Lotfi, E. Zysk, F. L. Siena, M. M. Carthy, and P. Breedon, “Parkinson’s disease: current assessment methods and wearable devices for evaluation of movement disorder motor symptoms - a patient and healthcare professional perspective,” *BMC Neurol.*, vol. 20, no. 1, pp. 1–13, 2020, doi: 10.1186/s12883-020-01996-7.

[6] A. T. Tzallas *et al.*, “Perform: A system for monitoring, Assessment and management of patients with Parkinson’s disease,” *Sensors (Switzerland)*, vol. 14, no. 11, pp. 21329–21357, 2014, doi: 10.3390/s141121329.

[7] L. Sigcha *et al.*, “Deep learning approaches for detecting freezing of gait in parkinson’s disease patients through on-body acceleration sensors,” *Sensors (Switzerland)*, vol. 20, no. 7, 2020, doi: 10.3390/s20071895.

[8] M. Mancini *et al.*, “Measuring freezing of gait during daily-life: an open-source, wearable sensors approach,” *J. Neuroeng. Rehabil.*, vol. 18, no. 1, pp. 1–13, 2021, doi: 10.1186/s12984-020-00774-3.

[9] M. Bächlin *et al.*, “Wearable assistant for Parkinsons disease patients with the freezing of gait symptom,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 2, pp. 436–446, 2010, doi: 10.1109/TITB.2009.2036165.

[10] A. Ferrari, P. Ginis, M. Hardegger, F. Casamassima, L. Rocchi, and L. Chiari, “A mobile Kalman-filter based solution for the real-time estimation of spatio-temporal gait parameters,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 24, no. 7, pp. 764–773, 2016, doi: 10.1109/TNSRE.2015.2457511.

[11] C. Tunca, N. Pehlivan, N. Ak, B. Arnrich, G. Salur, and C. Ersoy, “Inertial sensor-based robust gait analysis in non-hospital settings for neurological disorders,” *Sensors (Switzerland)*, vol. 17, no. 4, pp. 1–29, 2017, doi: 10.3390/s17040825.

[12] J. Camps *et al.*, “Deep learning for freezing of gait detection in Parkinson’s disease patients in their homes using a waist-worn inertial measurement unit,” *Knowledge-Based Syst.*, vol. 139, pp. 119–131, 2018, doi: 10.1016/j.knosys.2017.10.017.

[13] A. Zhao, L. Qi, J. Li, J. Dong, and H. Yu, “A hybrid spatio-temporal model for detection and severity rating of Parkinson’s disease from gait data,” *Neurocomputing*, vol. 315, pp. 1–8, 2018, doi: 10.1016/j.neucom.2018.03.032.

[14] H. Yin and N. K. Jha, “A Health Decision Support System for Disease Diagnosis Based on Wearable Medical Sensors and Machine Learning Ensembles,” *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 3, no. 4, pp. 228–241, 2017, doi: 10.1109/TMSCS.2017.2710194.

[15] A. J. Espay *et al.*, “Technology in Parkinson’s disease: Challenges and opportunities,” *Mov. Disord.*, vol. 31, no. 9, pp. 1272–1282, 2016, doi: 10.1002/mds.26642.

[16] A. L. Silva de Lima *et al.*, “Home-based monitoring of falls using wearable sensors in Parkinson’s disease,” *Mov. Disord.*, vol. 35, no. 1, pp. 109–115, 2020, doi: 10.1002/mds.27830.

[17] A. L. S. De Lima *et al.*, “Feasibility of large-scale deployment of multiple wearable sensors in Parkinson’s disease,” *PLoS One*, vol. 12, no. 12, pp. 1–15, 2017, doi: 10.1371/journal.pone.0189161.

[18] M. Hammouda, “A study on Multi-axis sensors and Machine Learning system for gait analysis in security and healthcare.” 2020.

[19] S. M. Keloth, R. Viswanathan, B. Jelfs, S. Arjunan, S. Raghav, and D. Kumar, “Which gait parameters and walking patterns show the significant differences between Parkinson’s disease and healthy participants?,” *Biosensors*, vol. 9, no. 2, 2019, doi: 10.3390/bios9020059.

[20] K. J. Kubota, J. A. Chen, and M. A. Little, “Machine learning for large-scale wearable sensor data in Parkinson’s disease: Concepts, promises, pitfalls, and futures,” *Mov. Disord.*, vol. 31, no. 9, pp. 1314–1326, 2016, doi: 10.1002/mds.26693.

[21] N. M. Rad, T. van Laarhoven, C. Furlanello, and E. Marchiori, “Novelty detection using deep normative modeling for imu-based abnormal movement monitoring in parkinson’s disease and autism spectrum disorders,” *Sensors (Switzerland)*, vol. 18, no. 10, 2018, doi: 10.3390/s18103533.

[22] R. San-Segundo, H. Navarro-Hellín, R. Torres-Sánchez, J. Hodgins, and F. de la Torre, “Increasing robustness in the detection of freezing of gait in Parkinson’s disease,” *Electron.*, vol. 8, no. 2, pp. 1–14, 2019, doi: 10.3390/electronics8020119.

[23] U. Harvard, “Applications of Tiny ML,” 2021. [Online]. Available: https://online-learning.harvard.edu/course/applications-tinyml?delta=0. [Accessed: 30-Jun-2021].

[24] “MJFF Levodopa Wearable Sensors Dataset.” 2015.

[25] A. Marcante *et al.*, “Foot pressure wearable sensors for freezing of gait detection in parkinson’s disease,” *Sensors (Switzerland)*, vol. 21, no. 1, pp. 1–12, 2021, doi: 10.3390/s21010128.

[26] C. Ricciardi *et al.*, “Classifying patients affected by Parkinson’s disease into freezers or non-freezers through machine learning,” *IEEE Med. Meas. Appl. MeMeA 2020 - Conf. Proc.*, 2020, doi: 10.1109/MeMeA49120.2020.9137317.

[27] A. Zhan *et al.*, “High Frequency Remote Monitoring of Parkinson’s Disease via Smartphone: Platform Overview and Medication Response Detection,” pp. 1–12, 2016.

[28] L. Heim, A. Biri, Z. Qu, and L. Thiele, “Measuring what Really Matters: Optimizing Neural Networks for TinyML,” 2021.

[29] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “snopes.com: Two-Striped Telamonia Spider,” *J. Artif. Intell. Res.*, vol. 16, no. Sept. 28, pp. 321–357, 2002.

[30] “TinyML.org Home Page,” 2021.

[31] B. Newman, “Enabling Ultra-Low Power Machine Learning at the Edge,” *Tiny ML EMEA 2021*, 2021. [Online]. Available: https://cms.tinyml.org/wp-content/uploads/emea2021/tinyMLEMEA2021d2Keynote\_Newman.pdf.

[32] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, pp. 1–14, 2016.

[33] DeepLearning.AI, “AI for Medical Diagnosis,” *Coursera Course*, 2020. [Online]. Available: https://www.coursera.org/learn/ai-for-medical-diagnosis.

[34] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller, “Deep learning for time series classification: a review,” *Data Min. Knowl. Discov.*, vol. 33, no. 4, pp. 917–963, 2019, doi: 10.1007/s10618-019-00619-1.

[35] U. Harvard, “Fundamentals of TinyML.” [Online]. Available: https://online-learning.harvard.edu/course/fundamentals-tinyml?delta=0. [Accessed: 18-Jun-2021].

[36] U. Harvard, “Deploying Tiny ML.” [Online]. Available: https://online-learning.harvard.edu/course/deploying-tinyml?delta=0.

# Appendices

## Appendix A – Engineering Log Book

The engineering log book for this project was maintained between February and August 2021, to log discussions with a supervisor, decisions and actions.

|  |  |  |
| --- | --- | --- |
| **Date** | **Discussion topics** | **Decisions and Actions** |
| 19-Feb-2021 | Call with Prof Darwazeh  Reviewed data sources and datasets available for PD  Discussed focus of research  Agreed to have a regular call for the next 6 weeks to finalise focus and scope of the study | Actions:   * Request ID letter to obtain for Synapse MJFF Levodopa dataset * Focus on identifying the data sets and ML focus * Share several key research papers |
| 26-Feb-21 | Call with Prof Darwazeh | * Reviewed Parkinson’s Disease datasets available for research; follow up with exploratory data analysis. |
| 26 Mar-21 | Call with Prof Darwazeh  Reviewed summary of research into ML vs Statistical analysis, pros and cons  Updated Project registration form and risk assessment form  Feedback:   * Make sure to define all concepts, e.g., overfitting * Need to multiplex the areas of work: research, practical with sensors, ML modelling and analysis * Plan: in 6 weeks start looking at implementation * Background study: which method and why (non-ML vs ML vs DL). Convince the reader why one or the other, compare performance. * Build the sensor, consider if data collection is possible. If not, use the public dataset to develop the model | * Submit two report forms (deadline 29Mar) * Update project title to “Machine learning approach for Parkinson disease monitoring using wearable technologies” * Plan: in 6 weeks start looking at implementation. Complete lit study section by 30 April (deadline for Literature Review report) * Next call: 2 April, 11am |
| 2-Apr-2021 | Update call  Shared Literature Review report draft. Discussed overall focus of the study – several potential areas include deep learning for PD – is it fit for purpose? | Agreed to look at the intersection of ML and sensors, which is deep learning, supervised ML and sensors (TinyML is the framework that allows to load ML models on microcontrollers). |
| 9-Apr-2021 | Prepared brief summary of ML signal processing and statistical methods for PD | Next steps: prepare overview of PD use cases, including prognosis, diagnosis and medical treatment. |
| 23-Apr-2021 | Prepared brief summary of PD use cases | Next steps: prepare overview of wireless sensors and their use for PD. |
| 18-May-2021 | Prepared brief summary of wireless sensor use cases |  |
| 31-May -2021 | Prepared presentation for the discussion with Prof. Schrag, to validate the focus of the study | Prepare a questionnaire for discussion with Prof. Schrag. |
| 28-Jun-2021 | Call with Prof’s Darwazeh, Prof. Sally Day and Prof. Anette Schrag  Presented focus of the ML for PD study.  Discussed the questionnaire attached in Appendix B of the report. | Good feedback on the focus of the study from Prof. Schrag.  Agreed to share any questions that need clarification from the PD practitioner’s perspective |
| June-2021 | Audited online Harvard course “Fundamentals of TinyML” on Coursera. | Understood principles of machine learning workflow, learned fundamentals of Tiny ML and TensorFlow Lite/ Lite Micro as one framework applying principles of Tiny ML |
| June-July-2021 | Audited online Harvard course “Applications of TinyML” on Coursera. | Reviewed and applied several Tiny ML applications, including Magic Wand, which uses accelerometer readings. |
| 8-July-2021 | Call with Prof. Day and PhD student Yanke Sun who is starting to work on PD research.  Presented focus of the ML for PD study; discussed availability of datasets. | Agreed to share synthetic data created for finger tapping; aim to use this dataset to test deep learning models post-MSc report. |
| July 2021 | Audited online Harvard course Deploying TinyML on Coursera | Learned how to deploy TensorFlow Lite Micro models on microcontrollers. |
| 17-Aug-2021 | The deep learning models – convolutional and recurring neural networks, and a supervised – Random Forest model – were trained on the full dataset. The results are quite similar – all three show ~97% accuracy but classification of smaller classes is not accurate. Trained for sensitivity / specificity.  On the model size – the compression for convolutional deep learning model works very well, the size goes down to below 10KB, which is comparable with examples of supervised model sizes reported in other reports. Compression for LSTM is under 100KB, which is too big for microcontrollers.  Inference performance on the microcontroller has not yet been tested. | Section 7 – analysis of the results – and Section 8 – Conclusions and Next Steps – are in progress.  Next step – review misclassified labels; re-run the models with class weighting.  With these results, the initial hypothesis – that DL performs better than supervised models – needs further testing.  Performance of the supervised model on one patient has very similar accuracy results. The next step is to check how well the supervised algorithm will generalise on the full dataset, and how well it predicts different classes. |
| 21-Aug-21 | Call with Prof Darwazeh to discuss final results of the study. | Finalise Results and Conclusions sections  Update Appendices  Add Executive Summary |
| 27-Aug-21 | Final report draft shared with Prof Darwazeh | Final edits on the report |
| 1-Sept-21 | Report copies submitted |  |

## Appendix B – Link to GitHub Project Depository

The below link is for GitHub repository, which hosts Jupyter Notebooks with the Code developed for the project:

<https://github.com/sgrantcs/MLforPDFinalReport/>

## Appendix C - Neurologist Discussion Questionnaire

The discussion with Professor Anette Schrag took place on 28 June to discuss the approach to the machine learning modelling for Parkinson’s Disease. The following questionnaire was shared with Prof. Schrag prior to the discussion:

Questions related to datasets:

* Only 12 patients have been included in the MJFF Levodopa Study. Are 12 patients enough to draw conclusions from wearables studies?
* Is accelerometer data enough? Do we need to look at the gyroscope data + combination of other sensors (e.g., perspiration, temperature)
* Are 3 phenotypes enough (e.g., tremor, dyskinesia, bradykinesia)?

Questions related to PD use cases and applications:

* Which use cases that use wearable sensors / ML are of most interest to PD practitioners?
* Which format should the data be presented in?
* Is there interest / possibility in setting up a trial to test any use cases?

Questions related to Machine Learning for PD:

* What accuracy levels provide enough confidence to adopt wearables for decision making?
* What is needed to build confidence in ML:
  + More studies/ bigger size of studies?
  + Visualisation?
  + Model transparency & explainability vs accuracy?
  + Other

Questions related to the Wearables Studies:

* Who are the main target audience for remote monitoring applications? Patients? PD neurologists?
* What data would be most useful for patients?
* What data would be most useful for PD neurologists?
* What are the barriers to adoption of wearable – based monitoring today?

## Appendix D - Neurologist Discussion Summary

The discussion with Professor of Clinical Neurosciences Anette Schrag took place on 28 June 2021 to discuss the approach to the machine learning modelling for Parkinson’s Disease. This was followed by a discussion with Prof. Sally Day who is also overseeing a Healthcare Wearables study.

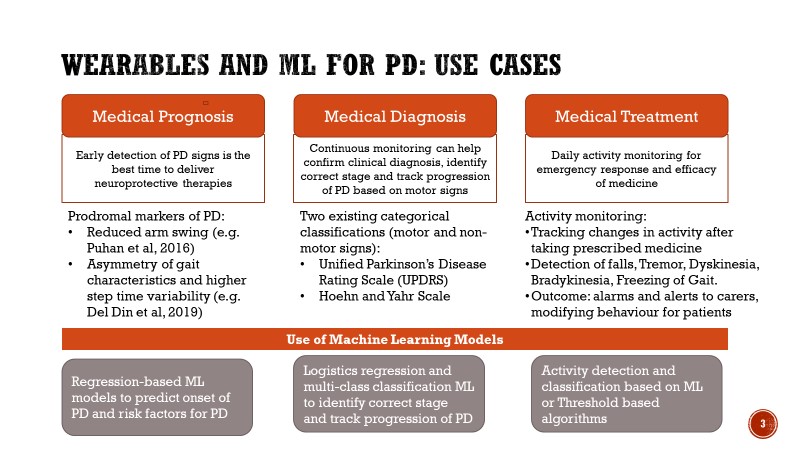
The summary of the discussion, which focussed on the data for PD, is as follows:

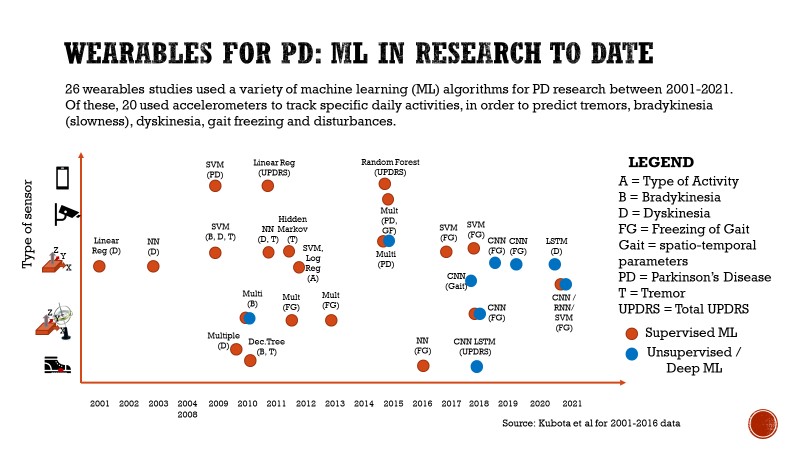
* Small size of the MJFF dataset will not allow to generalise the conclusions of this study and will require further exploration on a large sample. Ideally, a study would include both PD patients and healthy individuals as controls.
* General availability of large-scale data is a challenge: In another PD study currently supervised by Prof, simulated data is used to help identify features.
* Medical treatment is the most interesting area for exploration. There are some challenges around Medical Diagnosis use cases, such as using ML for predicting UPDRS, which includes both motor and non-motor evaluation.
* Overall positive feedback was given on the focus of the study; There is interest in using ML for remote monitoring of motor symptoms in the community of PD researchers. Prof Schrag suggested to focus on classifying one symptom at a time. Accelerometer data is enough for the modelling, even if a use of 6-axis or 9-axis IMU would provide additional sensor data.

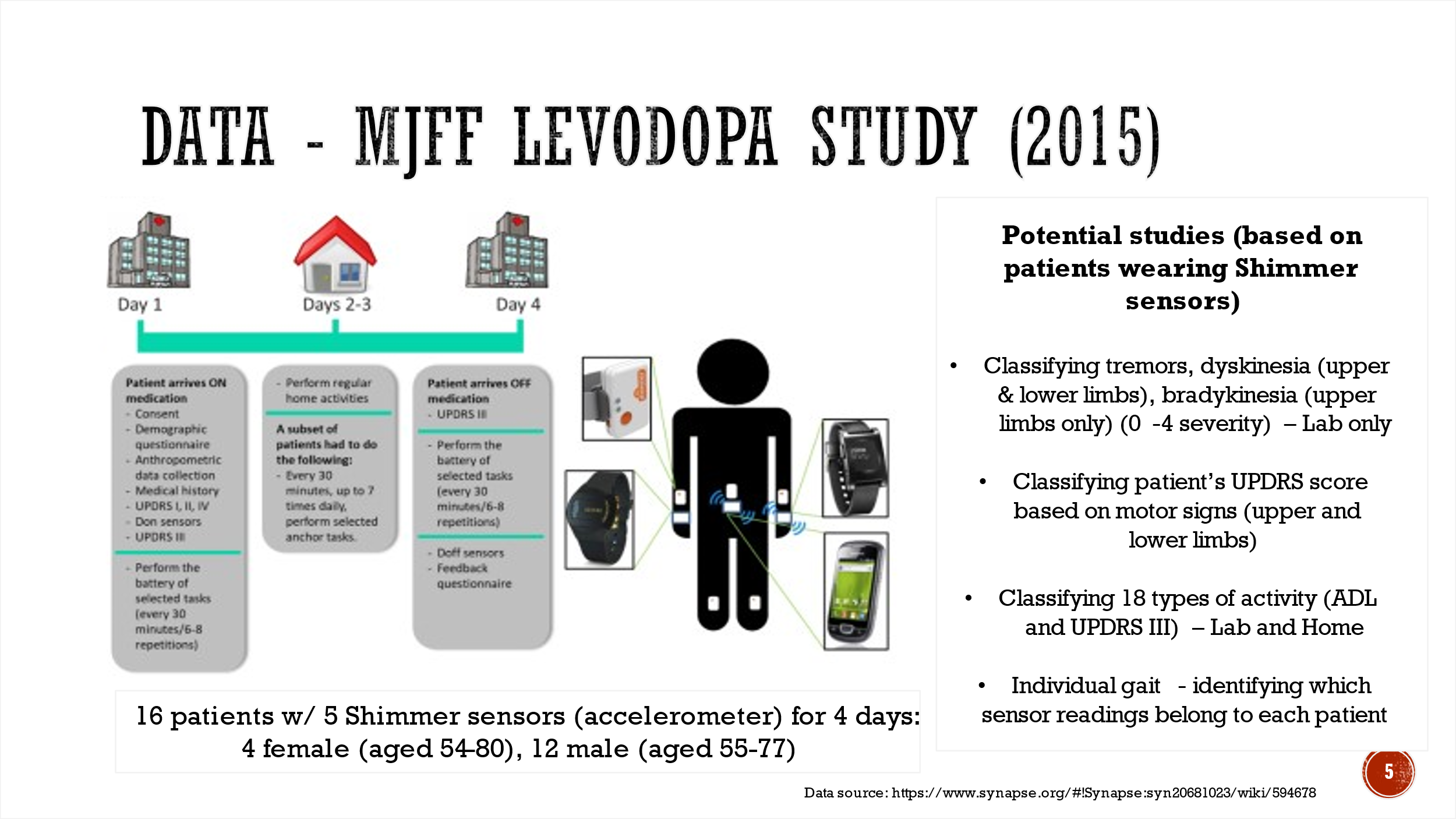
The following presentation was shared during the discussion:

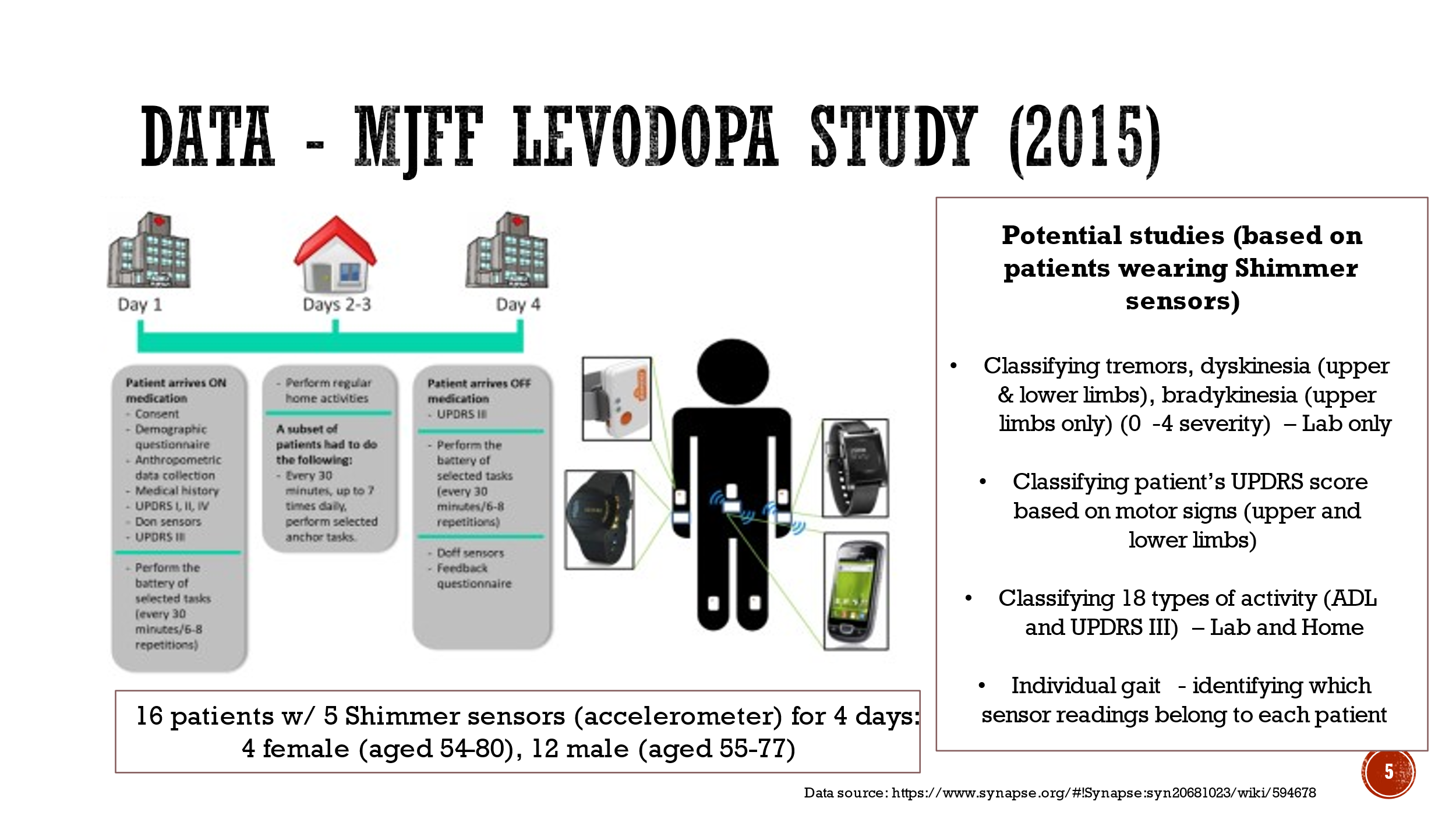


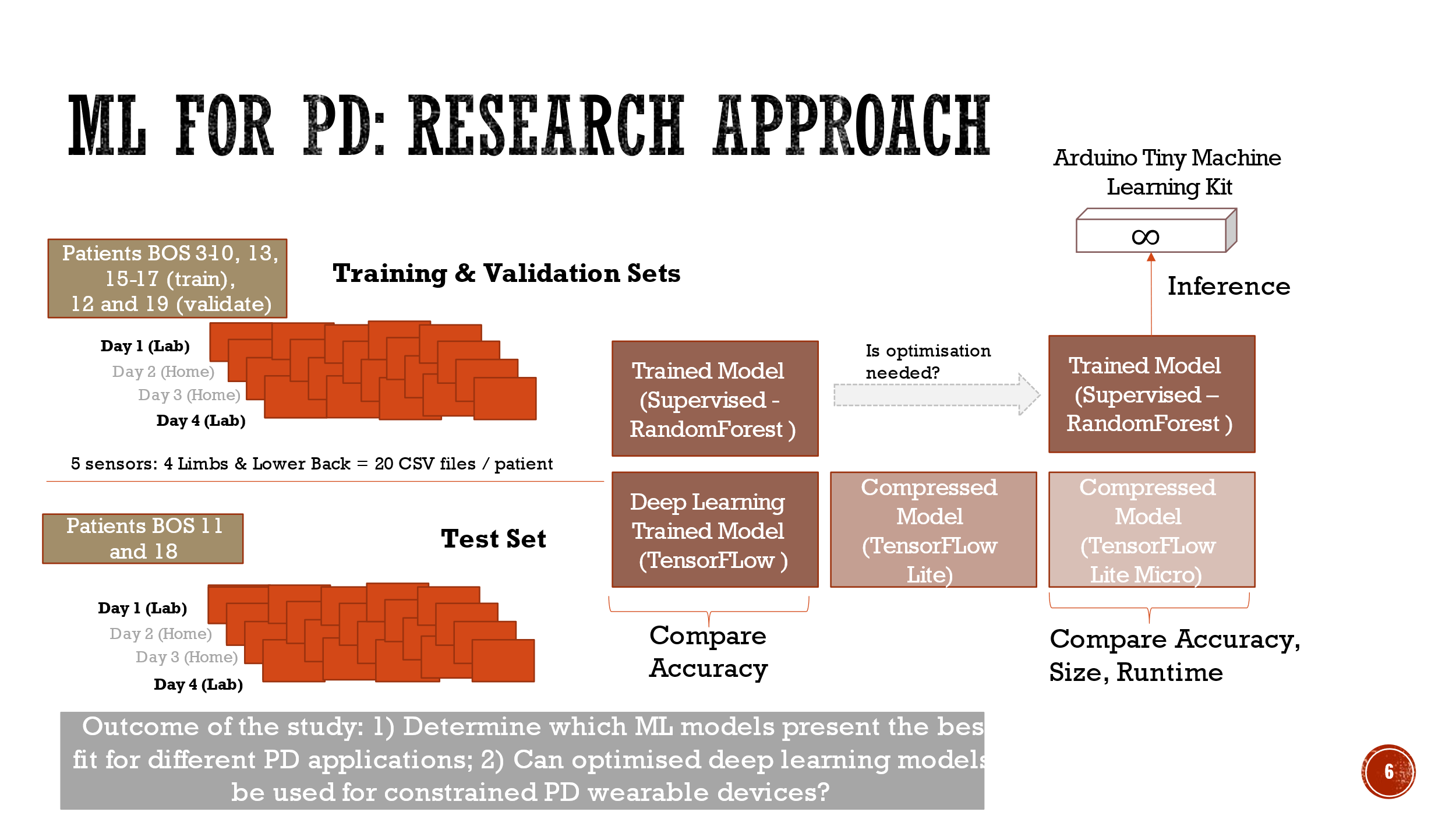


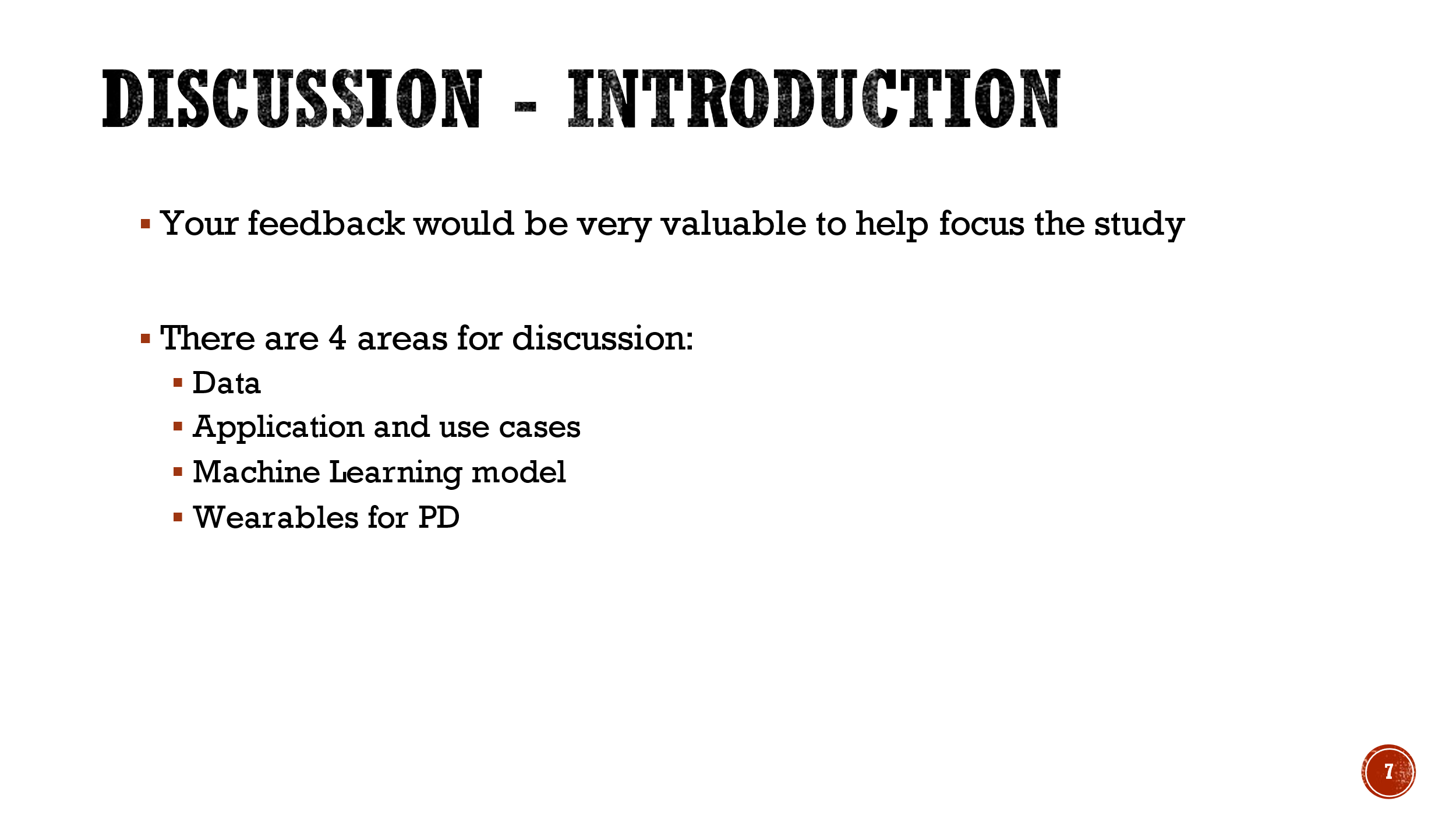












1. **Code and commit history for the project is available on GitHub:** <https://github.com/sgrantcs/MLforPDFinalReport/> [↑](#footnote-ref-1)