Journal of Intelligent & Robotic Systems (2025) 111:32
https://doi.org/10.1007/s10846-025-02226-3

REGULAR PAPER

Development of a Microcontroller-Based Recurrent Neural Network Predictive System for Lower Limb Exoskeletons

T. Slucock[1] · G. Howells[1,2] · S. Hoque[1] · K. Sirlantzis[3]

Received: 29 October 2023 / Accepted: 3 January 2025

Abstract
Practical deployments of exoskeletons can often be limited by cost, limiting access to their usage by those that would benefit from them. Minimising cost whilst not harming effectiveness is therefore desirable for exoskeleton development. For Control Systems governing assistive and rehabilitative exoskeletons that react to the wearer's movements, there will inevitably be some delay between when their wearer intends to move and when the exoskeleton can assist with this movement. This can lead to situations where a user may be limited by their own assistive exoskeleton, reducing their ability to move freely. A potential solution to this is to provide a proactive method of control, where the most likely path of the wearer's movement is predicted ahead of the wearer making the motion themselves. This can be used to give the user assistance immediately as they are walking, as well as potentially pre-emptively adjust their gait if they suffer from predictable gait deficiencies. The purpose of this paper is to investigate the Data Collection, Implementation, and Effectiveness of an LSTM Recurrent Neural Network dynamically predicting future movement based off of prior movement. These methods were developed to use off the shelf, Low-Cost Microcontrollers as to minimise their Financial, Weight, and Power Impact on an overall Low-Cost exoskeleton design, as well as to evaluate how effective such an implementation would be when compared to running such a Neural Network on a more powerful processor. The created model was capable of achieving similar accuracies to far more powerful models on High-Powered Laptops.

Keywords Assistive devices · Lower limb exoskeleton · Microcontrollers · Neural networks · Wearable robots

1 Introduction

The concept of an actuated leg attachment designed to aid in walking can be traced back to as early as 1890 [1], the first working examples are seen in 1969 with Mihajlo Pupin Institute's "Kinematic Walker" [2] and in 1971 with General Electric's "Hardiman I" [3]. These two active exoskeletons, actuated by powered electric motors as opposed to passive force, required governing control systems to determine how, how much, and when their component actuators would move, governed by input sensor data. Neural Networks are one such control system implementation.

As of 2022, the average UK medical spending per capita was estimated at $4,192 per person [4]. By contrast, exoskeletons such as EksoNR could cost as much as £126,000 [5], a value infeasible for many to afford without external funding even amongst wealthier countries. Within those nations of lower average medical spending and individual wealth acquiring such equipment becomes effectively impossible for the average person without external funding. An alternative method seen in rehabilitation centres therefore is to schedule Rehabilitation Sessions, where a user suffering from gait degradation may have temporary access to an advanced rehabilitation exoskeletons where they may aid in improving their gait patterns. These sessions have shown to be effective [6], however suffer from limitations such as patient proximity, and delay between sessions.

The supervisors contributed equally to the paper.

✉ T. Slucock
tomslucock@gmail.com

[1] School of Engineering, University of Kent, Giles Lane, Canterbury CT2 7NT, England

[2] School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, England

[3] School of Engineering, Technology, and Design, Canterbury Christ Church University, N Holmes Rd, Canterbury CT1 1QU, England

Published online: 26 February 2025

This paper focuses primarily on exoskeletons of a rehabilitative or assistive focus, designed to be used by wearers suffering pathologic gait issues such as Hypo/Hypertonia (muscle limpness/stiffness) and Hypo/Hyperkinesia (poor/excessive muscle movements) but are none the less capable of some independent movement such as sufferers of Cerebral Palsy or Stokes. This paper focuses on making an AI Predictive System, sees how small this system can be made, and whether it maintains effectiveness. In further detail: a) the paper develops an AI Prediction System that may aid in a control system's ability to determine the likely future motion of a wearer, by pre-emptively assisting this motion, and assesses its effectiveness based on the percentage error it predicts from reality, b) it seeks to discover how small and inexpensive of a profile such an AI predictive system can be fit onto whilst maintaining effectiveness, such as a commercially available microcontroller. Due to the high cost of many currently existent methods for exoskeletons it would be beneficial to focus on reducing this cost such as through simpler components and implementations, for example, as detailed in [7] there is a need for Lower Cost Exoskeletons able to more readily aid those with Gait Deficiencies. Inexpensive exoskeletons may allow for their wider deployment by expanding the range of individuals capable of affording them, or centres capable of accessing them.

The proposed implementation takes the form of a Recurrent Neural Network (RNN), specifically a Long-Short Term Memory (LSTM) prediction system designed to predict future knee angle movement of the wearer using prior movement such that a theoretical assistive exoskeleton could then aid in these movements or attempt to correct potential deficiencies by moving to these predicted knee angles as the wearer does, thereby aiding in their movement. With LSTM benefitting from using memory of prior events to inform future predictions, which is appropriate for predicting repetitive walking as prior strides can inform later ones. [8] also predicts knee angle, although in simulation. LSTMs also possess superior long-term memory, reducing the rate at which learned information attenuates in its ability to inform decision making [9]. LSTM's have also been proven to be capable of effectively running on the Cortex-M Microprocessors used by many inexpensive microcontrollers [10]. Being directly supported by TensorFlow, a well-established Machine Learning API, with TensorFlow Lite being used in this implementation.

2 State of the Art

As Described by Caldas R et al. in [11] up until 2016 some of the most common Machine learning Implementations within exoskeletons were either Artificial Neural Networks (ANN's) or Hidden Markov Models (HMM's). Common use cases were determining Gait Events such as heel strikes and toe off as well as swing/stance positions, Spatiotemporal Parameters such as Stride and Step/Step/Stance Times as well as Gait Velocity and Cadence, and Joint Angles which are most relevant to this paper [12]. Earlier examples of Joint Angle prediction and measurement using machine learning can be seen in [13] which made use of a General Regression Neural Network (GRNN) to estimate the Hip, Knee, and Ankle Angles based off of motion sensors at the hip, knee, and ankle joints. Meanwhile Sun et al. [14] describes other examples of Machine Learning up to 2020, such as Reinforcement Learning [15], Support Vector Machines [16], etc. Examples of Recurrent Neural Networks meanwhile are limited, as are using such models to predict joint angles. An earlier example of such an implementation from 2019 by Mundt et al. [17] simulated Accelerometer and Gyroscope data from more reliable Optoelectronic markers, using this as training data for a Feedforward Neural Network (FFNN) and a Long Short-Term Memory Neural Network (LSTM). Other examples have built on this, such as [18] [19].

Many LSTM Model examples run on high-power computers, laptops, or dedicated controllers, with regards to implementations on Microcontrollers [20] separately implemented both a Convolutional Neural Network (CNN) and a Multi-Layered Perceptron (MLP) onto an ESP32 Microcontroller, with a Shank Mounted IMU and current phase to act as input data to predict the angular velocity of the foot ~ 200 ms into the future. Many Previously described LSTM Models such as seen in [21] used multiple LSTM Hidden layers each of hundreds of units, whilst allowing for improved accuracy it also increases the size of the model and as Microcontrollers are limited for space a reduced model size is a necessity. In terms of papers that use Neural Networks to predict the wearer's knee angle, [22] used Shank and Thigh Inertial Measurement Unit (IMU) Data provided to an ANN model to predict walking speed and knee angle, whilst [23] used IMU and Electromyography (EMG) data provided to an RNN model run on an STM32F4 Microcontroller (costing £16.40 [24]) to predict joint torque and moment, although did not use LSTM/GRU due to technological limitations. Finally, [25] used EMG and Knee angle data collected from sensors via a ATMega328p Microcontroller and processed on a laptop, similar to this paper's laptop implementation.

For Sensor usage, papers such as [26] make use of an LSTM provided data by Inertial Measurement Units (IMUs) placed on the lower leg to predict lower-limb joints, which alongside Potentiometers are commonly used, inexpensive sensors in exoskeletons.

When viewing effectiveness of microcontroller-based neural network predictions in terms of Average Percentage Difference from reality, [20] achieved an accuracy of ~ 3.75% with a CNN predicting 20–200 ms into the future. Meanwhile [23] achieved an accuracy of ~ 4% with an RNN predicting 50 ms into the future. This paper's LSTM network

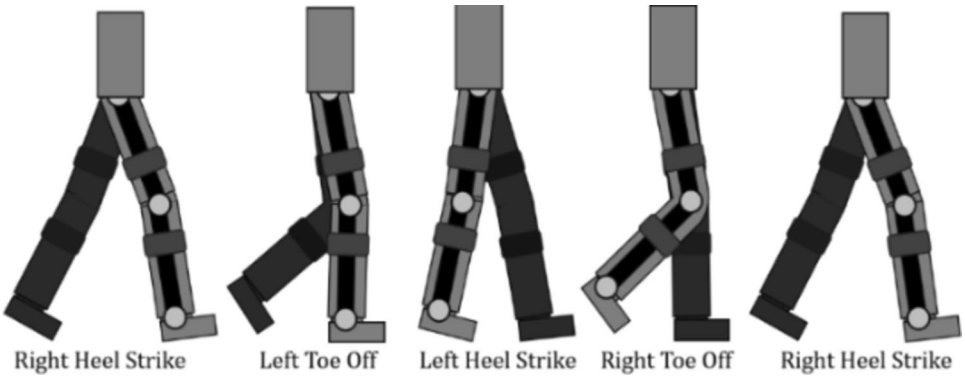

Fig. 1 Gait events over one cycle

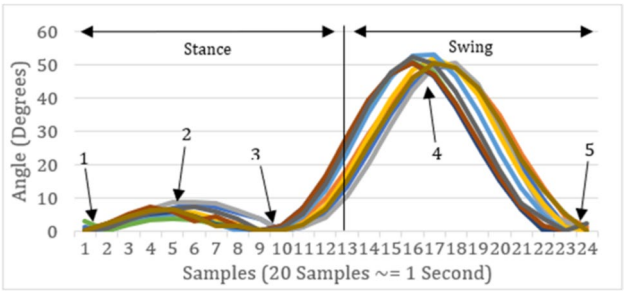

Fig. 2 Example of Repeated Gait Pattern. 1 Sample = 50 ms. 20 Hz Sampling Rate

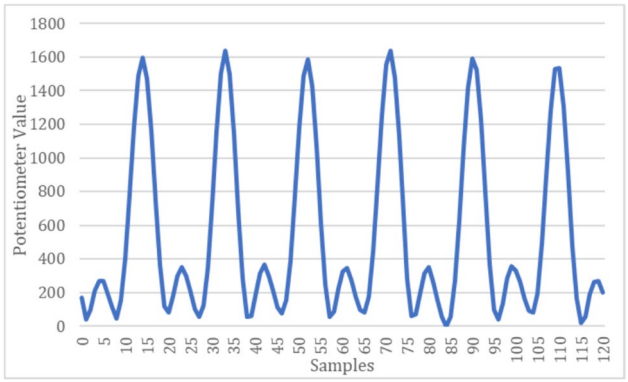

Fig. 3 Knee Angle change over 5 Gait Cycles, continuous. Each Cycle consisting of one Left and one Right Step. Left Knee, 20 Hz sampling rate

achieves similar accuracies with predictions made up to 1000 ms into the future.

3 Initial Overview

TensorFlow 2.11 and Keras 2.10 were used as the baseline for creating and training the Machine Learning Models using Python 3.10.9. This would then be converted to embedded C++/Arduino code to run on the two microcontrollers that were chosen for testing model effectiveness, The ESP32 and a Teensy 4.1 which were both chosen due to being inexpensive, possessing multiple cores for RTOS implementation, being Arduino code compatible, and being supported for TensorFlow Lite [27] Implementations. The AI recognition system would use the Time-Variant changes in knee angle and ankle motions as input data, to predict the wearer's knee angle in the future.

The Commonly Recognised Gait Events within most healthy individuals will follow the pattern seen in Fig. 1, Consisting of Right Heel Strike (1), Left Toe Off (2), Left Heel Strike (3), Right Heel Strike (4), and finally looping back to Right Heel Strike (5). The Knee Angle seen in Fig. 2 follows a consistent loop of Stance (1–3) and Swing (3–5) phases. Note: 0 Degrees represents standing straight.

Figure 3 displays 5 examples of walking cycle data collected at a 20 Hz sampling rate, as an average of sensor data collected at 100 Hz. The total time of the gait cycle was consistently 1.1–1.2 s, (~105 steps per minute) however walking rates amongst healthy adults vary dependant on age, health, and activity.

As defined in [28], movement, and walking in general can be split up into categories ranging from Incidental or Sporadic occasional movements whilst performing other tasks, to the Slow to Brisk walking speeds of intentional movements. The example in Fig. 3 is a set of Brisk Movement and the change in Knee Angle seen during it.

Recurrent Neural Networks are seen to be effective in predicting continuous, time variant data such as changes in knee angle, as seen in other papers such as [29] and [25]. Therefore being an effective choice for predicting these motions.

3.1 Hardware Implementation

Data Collection took the form of a simple knee orthosis constructed by the author, using a RS Pro P25 Potentiometer (10% Tolerance) to measure Knee Angle attached in line with the knee and an LSM9DS1 Inertial Measurement Unit (0.244 mg/LSB – milli-G's per Least Significant Bit) attached to the outer side of the ankle to provide

Accelerometer and Gyroscopic Data, the use of a single Inertial Measurement unit just above the ankle is similar to examples seen in [30] [31]. As the pattern of sensor movement was more important than exact values, sensor precision was not prioritised in favour if furthering a low-cost implementation.

This device was securely attached to the left leg via a basic exoskeleton frame, data was collated by an Adafruit HUZZAH32 ESP32 Feather [32], which was then sent via a wired serial connection to a Laptop which would via a concurrently running Python Program save all received information to a.csv file at 5 min intervals or when instructed to finish via button on the orthosis. This ESP32 would also run the LSTM prediction system, with predictions sent along with sensor values to be received and stored for later review.

The 8 Collected Data Points would therefore be the Average Accelerometer X, Y, and Z, Average Gyroscope X, Y, and Z, the Potentiometer Value, and Average Rate of Change of Potentiometer value each recorded to two decimal places. All Average values were calculated over the average of the last 5 samples (0.25 s), as to reduce the effects of random noise and so External/Parametric Uncertainties. Data Collection would occur at a rate of 100 Hz, or one sample per 10 ms, with 5 samples averaged to form a 20 Hz true Input rate (one sample per 50 ms). While ~ 100–200 Hz is a sampling rate seen in several exoskeleton control implementations such as in [30] [33], excessive data collection speeds for this implementation would lead to bloated sample file sizes, reduced time for predictions to be made which would ideally update every sample, and necessitate larger input data snapshots for the prediction system in order to display change in movement.

This would in turn make these prediction systems larger and slower to run on limited systems, the 100- > 20 Hz averaged conversion therefore reduced sensor noise whilst keeping sample rates acceptably low**.**

As the System would be predicting the wearer's motion into the future, as long as the prediction it made was further into the future than the time it took to process the prediction and for a controlled actuator to begin moving to that location, the delay between user motion and exoskeleton motion would not be of concern. As the exoskeleton would be effectively reacting before the wearer made the movement. This would only apply in cases where the wearer was in the process of moving periodically, as sudden movement from a still position would not be predictable from joint angle alone until the recognition system had received enough inputs to recognise a pattern to predict. As this process would require actuator implementation, it is beyond the scope of this paper.

The exoskeleton would however experience some Internal Uncertainty in its predictions, as it was guided solely by information regarding the wearer and had no knowledge of the wearer's environment. For example, the wearer kicking a wall and stopping their foot suddenly would appear very similar.

3.2 Data Collection

As the system would ideally be able to universally recognise of gait movements, a diverse range of motion need to be recorded for training purposes. These were decided based on distinct "Movement States" that would make up everyday movements, specifically: *Walking, Crouching, Walking Up Stairs, Sitting, Standing, Wandering,* and *Resting*. Within these, variations in speed and wearer would improve data diversity, as a recognition system is naturally incapable of "implying" information, if solely trained on walking data it may predict the user to be walking even in situations where they are standing still, as this is the only information the system "knows". This is seen in Fig. 4 where a Recurrent Neural Network was trained exclusively on constant walking data, then presented with a wearer in a static standing position. Note that a Potentiometer value of 0 represents standing

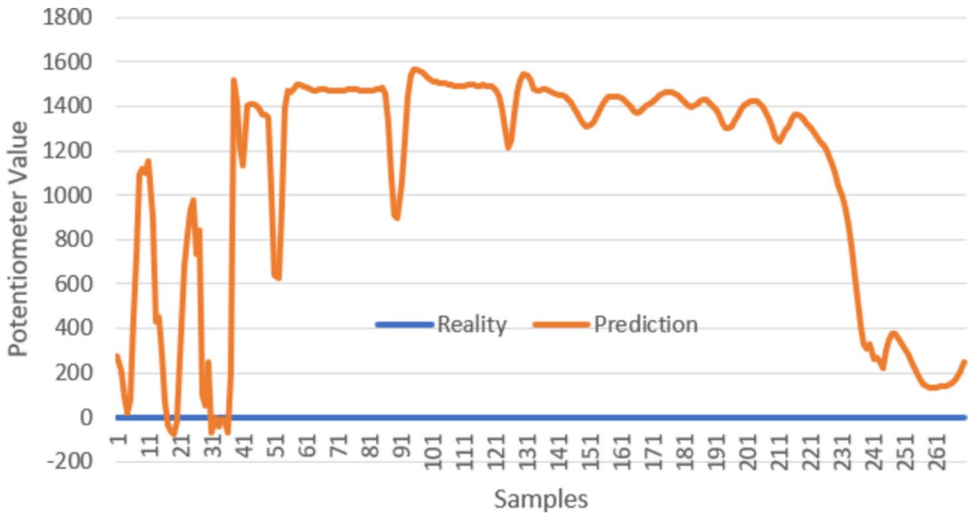

Fig. 4 Prediction System trained only on walking data struggles to predict the wearer standing still, as it has not been trained to "know" what standing still is. 20 Hz Sampling Rate

straight. Movement data for these states was collected from 12 volunteering healthy adults with an age range between 24 and 58, with 8 Male Candidates and 4 Female Candidates.

Datasets of ~ 5–10 min each were collected, in addition to further data being collected by the author by wearing the exoskeleton during extended walks.

Due to external circumstances, it was not feasible to collect data from individuals suffering from gait degradation. Instead, healthy patients would provide a proof of concept for the prediction system's ability to predict movement into the future.

4 AI Prediction Systems

A Recurrent Neural Network (RNN) was chosen due to its ability to consider prior inputs in its prediction of the future as seen in [34] where a Recurrent Neural Network produced lower errors than comparable CNN, FCN (Fully Connected Network), and Naïve Implementations. It was made with TensorFlow 2.11.

4.1 Data Processing and Model Training

The Acquired model data for the Prediction system would provide a "Snapshot" of data consisting of the last 40 samples across each sensor (Accelerometer X, Y, and Z, Gyroscope X, Y, and Z, Potentiometer, and Potentiometer rate of change), representing the last 2 s of sensor data prior to the current point.

This Feature Vector would be stored as a [40x8] Array, Accelerometer, Gyroscope, and Potentiometer data would each be normalised separately to between −1 and 1 based off of known likely maximum and minimum values collected during training, which would be equal to the largest and smallest value of the training set for each sensor that was within three standard deviations of the average as to prune anomalous spikes that would otherwise bias the dataset.

This snapshot process was completed for all input samples, totalling 215,000 samples (~ 3 h) of sample data which was later reduced to 58,000 samples to reduce bias as explained in 4.4. This was then randomly shuffled and split 90/10 to Training and Testing data.

Training data would train the model for 15 Epochs, which consisted of a [40x8] Input, and finished with a [30x1] output Dense Layer. A key benefit of both collecting potentiometer data and it being the value predicted by the Prediction system was that for testing purposes predictions could be directly compared with the "reality" of the actual sensor values that occurred at those predicted times to see how accurate the predictions were.

This process could be made to occur in real time to allow for prediction improvement via linear regression. The theoretical implementation achieved evaluated Mean Average Error of 4%. Linear Regression (Fig. 5) showed a strong association between predicted values and real values, which in turn displayed the prediction system as capable of recognising patterns within its input samples and accurately outputting predictions similar to reality. Predictions of future potentiometer values created from test data that the model was not trained were recognisably similar to the actual potentiometer values, although predictions tended to exaggerate reality. This was later found to be due to having too much sample data from a single source, drowning out variation from other sources (4.4). After Linear Regression, values tended to look more similar to reality (Fig. 6). Therefore, Linear Regression could potentially reduce prediction systematic errors.

While there was a small proportion of notable erroneous predictions, as each prediction is made independently of the previous, and predictions would be re-calculated every 50 ms, it would require a cascade of errors to threaten the user's walking capabilities. In such a case, it would be the duty of a lower-level control system layer to keep the position and direction of the exoskeleton and user within a safe range of each other.

Linear Regression was most effective when only trained on the 100 samples prior to the predicted samples, as opposed to training a regression model for the entire test dataset. This may be due to the predictions tending to vary in how much they over- and under-estimate reality over the length of a gait cycle or full data set, and so a smaller, more immediate subset for Linear Regression was better able to counteract the errors of the prediction by being more

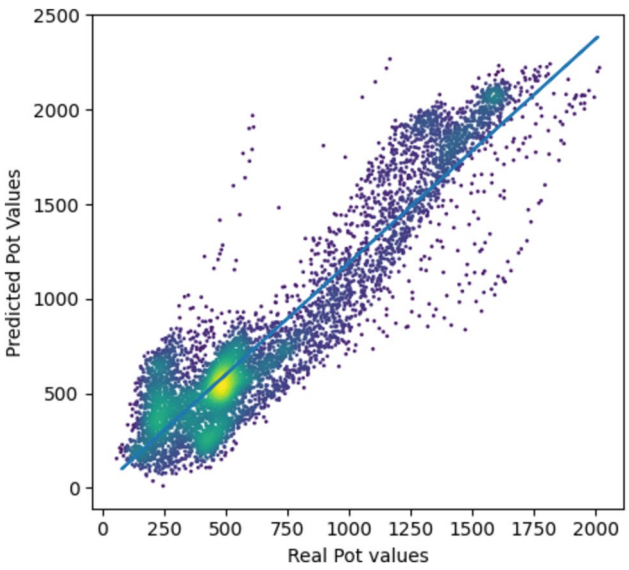

Fig. 5 Linear Regression comparing real potentiometer values to predictions made 10 samples (0.5 seconds) into the future. Lighter spots represent a higher density of values. $R^2 = 0.857$

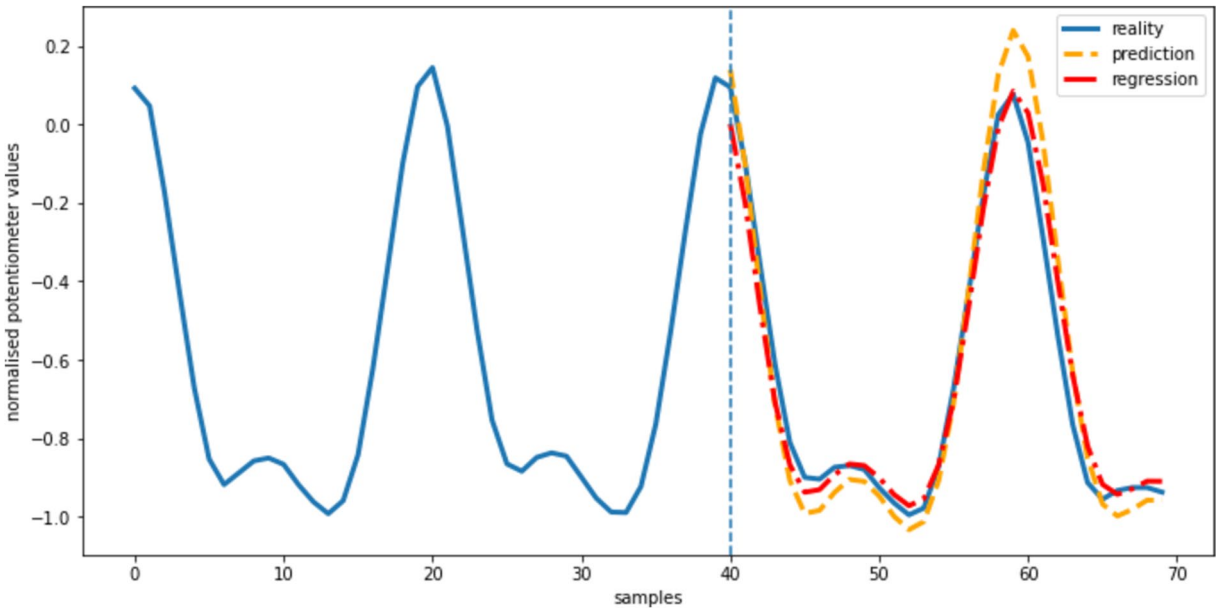

Fig. 6 Prediction (Orange) and Post-Regression Prediction (Red) vs Reality (Blue). Dividing line separates input data and output data

relevant to the present. The Laptop-based and Microcontroller-based implementations would be created using this initial design. The Laptop-based version having access to higher processing power made use of a larger, more powerful model, and therefore test the difference in effectiveness between it and the smaller microcontroller.

4.2 Laptop RNN Implementation

To act as a comparison to the Microcontroller a prediction model was tested on a high-power laptop that received sensor data from a connected exoskeleton and returned prediction data via serial connection. This allowed for the usage of a much larger LSTM network and would act as a baseline for comparing a model designed to make full use of a computer's predictive capability versus a microcontroller, the Network consisted of 4 Size 128 LSTM Layers similar to [34], as well as a Dropout and output [30x1] Dense Layer. The Laptop was placed in a backpack, with a USB cable providing the serial connection between the ESP located on the exoskeleton and the Laptop. The ESP32 used basic Arduino-based *Serial.print()/Serial.read()* Functions to send sensor data and receive predictions to and from the Laptop, where a running Python program would read and collect it, storing it in a Pandas Dataframe. If the Laptop was set to only collect data rather than predict it, it would then send a dummy reply back to the ESP32 to confirm that data had been received and that more data was to be sent.

If the program was set to predict data it would prior to the beginning of sample-reading load a saved model that had been previously trained, upon receiving at least the minimum number of samples to form an [40x8] input (or 2 s of samples). It would then make use of the.*predict_on_batch()* TensorFlow Lite function to output a single prediction set of 20 values (50 ms to 1000 ms into the future) based on this single batch of input data. Which would be saved within the same Pandas Dataframe as the samples.

Linear Regression would be performed on all values by comparing the previous predictions made to the actual potentiometer-measured knee angle values that occurred at those predicted times, then applying this calculated regression value to the present predicted data to counteract the difference. Testing the effectiveness of the Laptop-based implementation was done in the same way as data was originally collected in 3.2 – Data Collection, by running the system in real time whilst an exoskeleton wearer was walking a set path and logging predictions.

4.2.1 Effectiveness of Laptop Implementation

As was expected the Laptop Implementation had a similar Mean Average Error and produced similar results to that of the freshly trained model run on a PC and had an average Processing time per prediction set of less than the 50 ms cycle time. When comparing reality to predictions and regression predictions, Mean Average Error between reality and prediction varied between 3.63% to 8.1% for predictions made 50 ms to 1000 ms into the future, or 3.68% to 9.71% for regressed predictions. Both examples were averaged over

5,000 samples, a short depiction of 120 samples is seen in Fig. 7.

While this laptop implementation was reasonably accurate in predicting future values, it was limited in requiring a high-powered laptop in order to function at speed. The device used, a ROG Zephyrus M16 GU603ZX costing ~£4,000 [35] was more powerful than required, although any device used would need to be capable of running TensorFlow in addition to requiring a wired connection and backpack to carry the laptop and its power supply. In comparison, the ESP32 used in the first Microcontroller implementation costed only £15.50, over 250× less expensive in addition to smaller size and power usage. It was noted that in practice, Linear Regression had a negligible effect on Mean Average Error. This may be as the system was already quite accurate, and so there were minimal systematic errors that Linear Regression was most effective at reducing.

4.3 Microcontroller RNN Implementation

The Process of implementing an LSTM was first applied onto an ESP32 (Adafruit Huzzah32 ESP32 Feather Board) which used the *TensorFlow Lite for Microcontrollers* library [36], specifically "tflite-micro-esp-examples" [37] due to ESP32 specific optimisations and ease of use. Prior to September 2022, rolled LSTMs were not fully supported, causing a "UNIDIRECTIONAL_SEQUENCE_LSTM" compatibility error when attempted. Unrolled LSTMs were considerably larger than rolled in terms of file size and were effectively non-functional due to these size limitations. As the limitations of rolled LSTMs has since been resolved, the rough size of the tflite model was 245 kb in size, as opposed to the 3.2 MB savedModel file used by the laptop. It consisted of two LSTM layers of size 64 and 32, plus one Dropout layer to reduce overfitting and finally a [20 × 1] Dense layer to format the output, which were values chosen towards the upper maximum of size limitations that could fit on the ESP32's limited memory.

Predictions would be made using the Tensorflow Lite for Microcontrollers Libraries and would make predictions based off a [40x8] input array of sensor values, to make 20 output potentiometer knee angle predictions. The Parameters of the Model were determined by trial and error to discover a model that was effective within the limitations of the microcontroller's maximum allowable size and reasonable processing times. For example, more than two LSTM layers of similar size resulted in the model unable to function.

4.3.1 Effectiveness of Microcontroller Implementation

The ESP32 possessed a 240 MHz dual-core processor, the lower of the two tested microcontrollers. Due to the Microcontroller's Limited Processing power, the initial model could not make predictions within the 50 ms cycle window. Instead, predictions would take an average of ~180 ms to process, or a 5.56 Hz prediction rate. This resulted in a Mean Average Error of 8.93% to 11.87% for predictions made 50 ms to 1000 ms into the future. These predictions contained notably stepped changes as seen in Fig. 8, as the system would retain the previous prediction across several cycles in the time it took to process the next prediction.

Using pre-captured sample data to such that all predictions could be made without skipping, an ideal Mean Average Error of 6.1% to 10.9% could be achieved which would simulate the result of the ESP32 possessing sufficient processing power to predict every cycle. Linear Regression was not attempted due to these limitations, and as such no additional corrections were made to predictions.

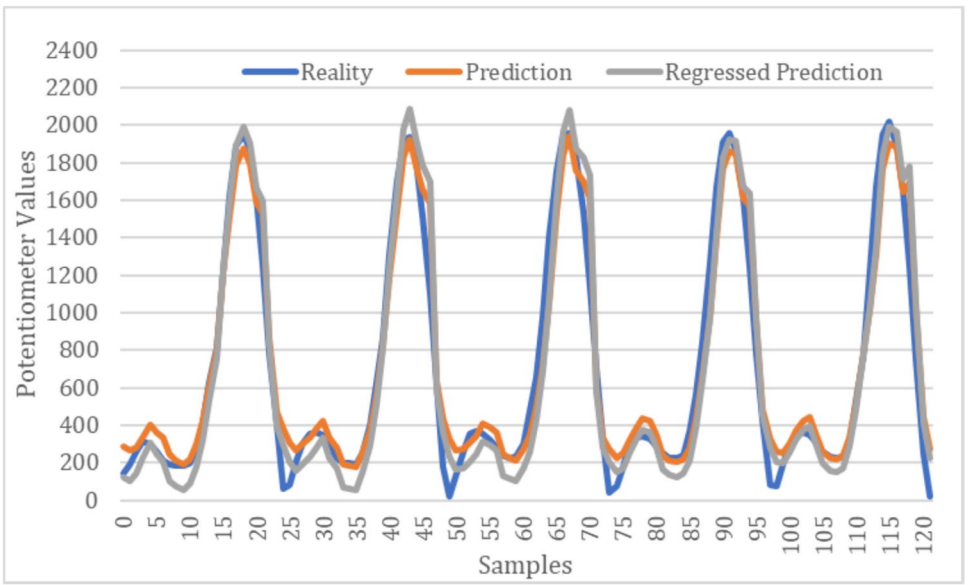

Fig. 7 Laptop Prediction vs Reality for data. Prediction made for 0.25 s into the future. 20 Hz Sampling Rate

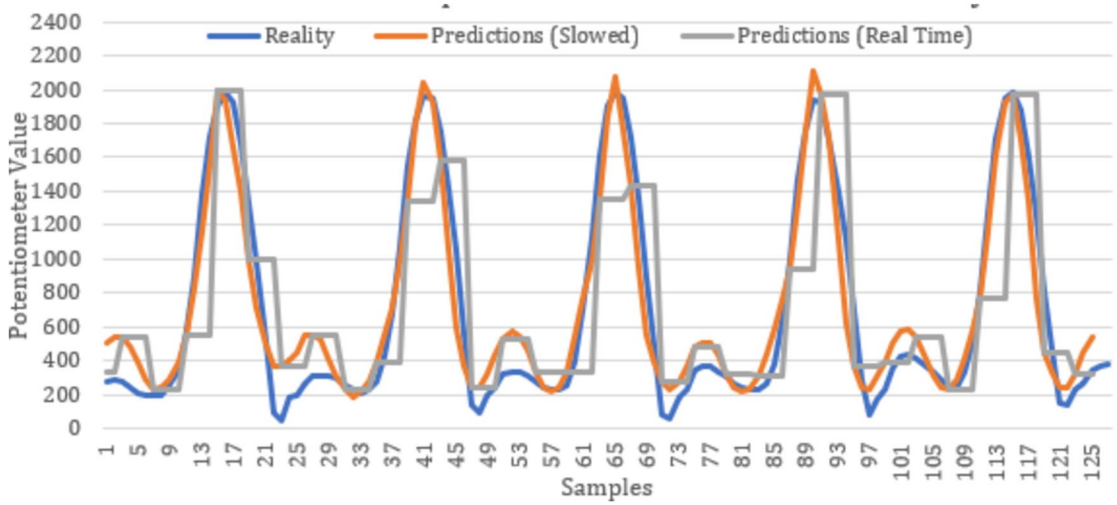

Fig. 8 Microcontroller Prediction vs Reality data. Prediction made 0.25 s into the future. 20 Hz Sampling Rate

4.4 Improving Microcontroller Implementation

The Primary Limitation of the Microcontroller implementation was its limited processing power, resulting in the stepped output when working in real time as a result of not being able to make predictions fast enough for the program. However, as seen by the "Slowed" output within Fig. 8, which was created using pre-captured sample data that was provided at the 5.56 Hz rate the system could handle, it was capable of producing accurate results. Therefore, it was considered worthwhile to investigate methods to speed up and otherwise improve the prediction system. The three methods considered were improving recognition system effectiveness, reducing the size of the input data array from [40x8] to a smaller value, and finally using a more powerful microcontroller.

4.4.1 Cleaning Sample Data

Whilst improving the recognition system does not aid in reducing processing time, improving its predictive capabilities and accuracy would allow increase functionality in spite of this slow processing time. From experimentation for both the Microcontroller and Laptop Implementations, using the full 3 h of collected data produced inferior accuracies as opposed to using a smaller subset of roughly 30–40 min of data.

The larger dataset's predictions made against test data were consistently less than their real values, in these cases the Linear Regression of the Laptop Implementation corrected for this (Figs. 9 and 10.

The likely reasoning for this was that most of this 3 h of data was collected by the author performing reletively

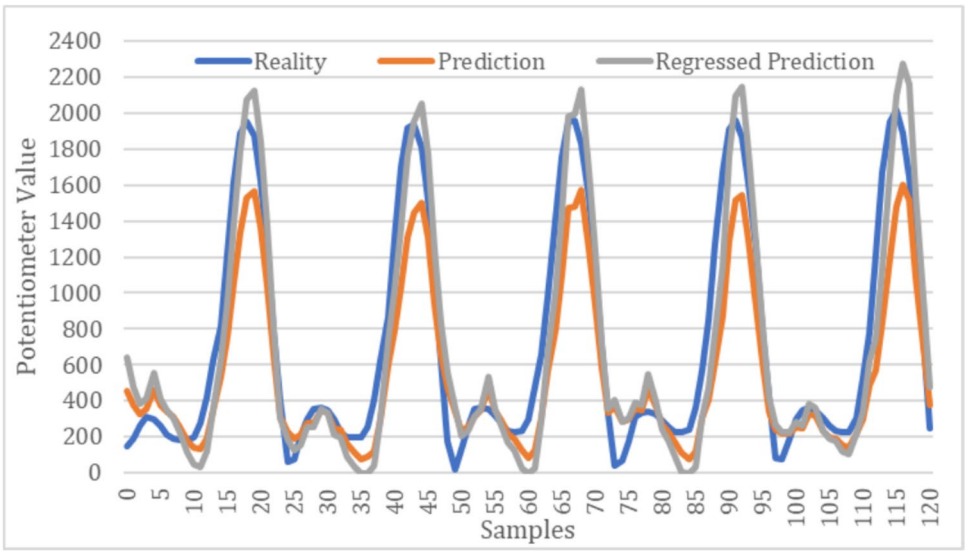

Fig. 9 Laptop Implementation using full dataset produced values that consistently underestimated reality, potentially overfitted. Prediction made for 0.25 s into the future. 20 Hz Sampling Rate

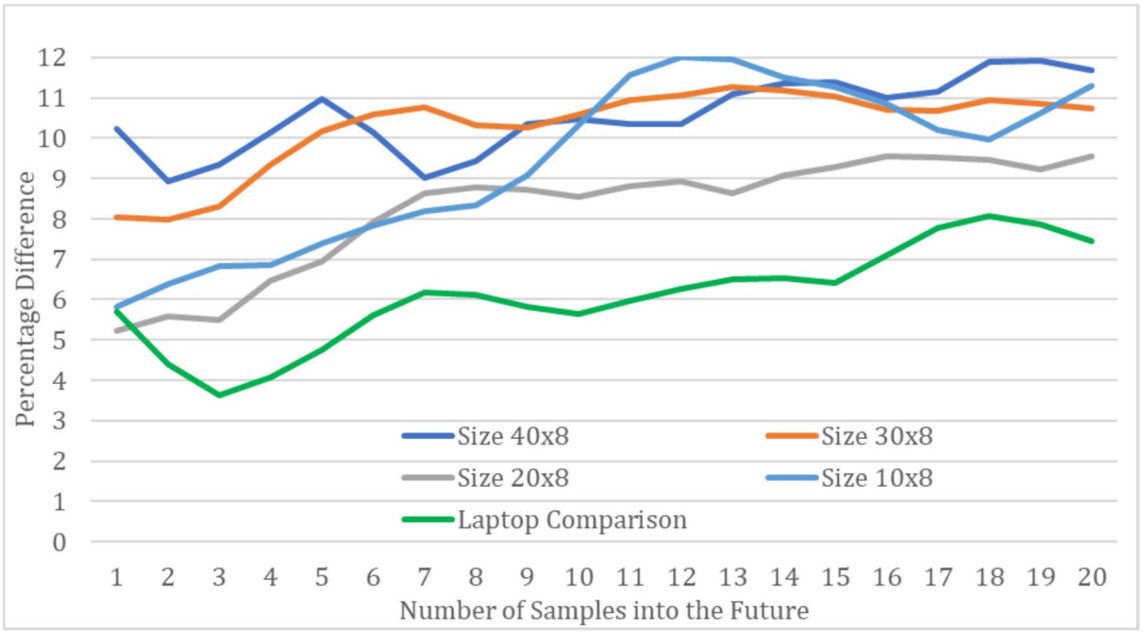

Fig. 10 Change in Percentage Difference over number of future predictions due to input size. 1 Sample = 50 ms

consistent walking, it was possible that this overwhelming majority of very consistent data polluted the dataset as a whole causing a notable bias due to Overfitting, by making any variety provided by test data from other participants insignificant in comparison. When corrected by removing most of this long-form walking data, as seen in Figs. 7 and 8, predictions were far more consistent with reality and showed less bias.

4.4.2 Reducing Input Size

It was found that while reducing the input size of the model did not affect model file size, it did have a proportional effect on the processing time of the model, with a smaller size resulting in a faster processing time. An increase in model size of 80 (10 × 8) resulted in a ~ 44 ms increase in processing time (Fig. 11), as such a range of models of different input sizes were run in real time to test the Mean Average Error of their predictions vs reality.

Table 1 shows the results of running each of these models for ~ 2 min each whilst walking the same path. The average percentage difference was calculated as the average of the percentage differences of all predictions made from 1–20 samples into the future for a particular input size, with these results shown in Fig. 10.

As seen by the results of Table 1, the [20x8] input size presents the lowest percentage difference, acting as the intermediate between having sufficient input size to make an accurate prediction and being able to do so at a quick

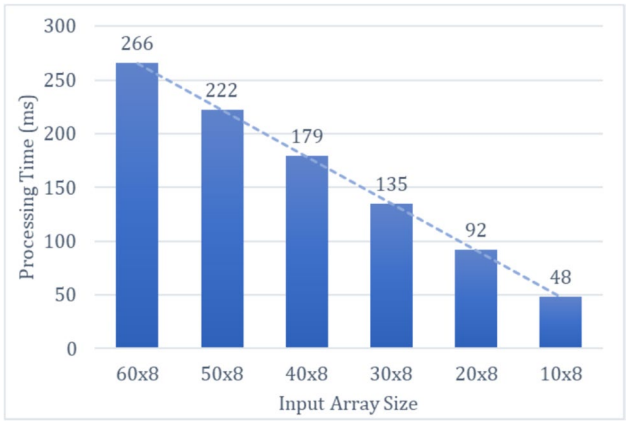

Fig. 11 Linear relationship between model input size and average processing time

enough speed to remain useful, this result is ~ 35% less accurate than the Laptop Implementation.

The higher errors, at [40x8] and [30x8], are likely of a lower accuracy due to the slower processing times (180 ms and 135 ms respectively) meaning that whilst having the most input data and therefore highest prediction accuracy, they cannot make predictions every 50 ms cycle, and therefore are not able to follow the continuously received data. [15x8] and [10x8] meanwhile see worse accuracy likely due to not having enough input data to make use of for an accurate prediction even if able to make predictions quickly.

Table 1 Mean Average Error for Microcontroller predictions

Input Size	[40x8]	[30x8]	[20x8]	[15x8]	[10x8]	Laptop (40×8)
Av. % Diff	10.56%	10.29%	**8.22%**	8.88%	9.41%	6.09%

4.4.3 Alternative Microcontroller Options

Finally, the second Microcontroller tested in this implementation was the Teensy 4.1. Possessing a Dual Core 600 MHz Cortex-M processor, 2.5 × the 240 MHz of the HUZZAH32 ESP32 Feather that had been used previously. This produced far faster results than expected, with a model of [40x8] input size having a processing time of ~ 35 ms, over 5 times faster than the ESP32, this was lower than the 50 ms program cycle time allowing the prediction system to run in real time, providing predictions every program cycle. This considerably improved the Percentage Difference of Predictions vs Reality. As such accuracy measurements were re-captured for 20, 40, and 60 sized input sizes to compare to the prior ESP32 experiments (Fig. 12). [40x8] input size proved the best overall percentage difference, varying from 2.01% to 5.01%, with an average of 3.85%, this result exceeded the accuracy of the Laptop Implementation. The Teensy 4.1 is more expensive than the ESP32 (~ £33.50 [38]), and of similar size.

The resultant Prediction vs Reality Graph seen in Fig. 13 shows that the Teensy 4.1's predictions were very similar to reality for average walking.

Due to the increase processing power, Linear Regression was implemented onto the Teensy for the [40x8] input, the results for average walking showed slightly inferior accuracies, likely as a result of the predictions already being very accurate with errors not being consistent enough for Linear Regression to have any benefit, however when tested for other situations such as standing or sitting still, linear regression proved far more effective, with the prediction system consistently over or underestimating knee angles by some amount and allowing linear regression to correct this continuous error.

The Test to find the ideal input size for the Teensy revealed an input of [40x8] produced the best Mean Average Error of ~ 3.85%, even capable of surpassing the significantly more powerful Laptop Model. This unexpected outcome prompted re-training the Laptop and Teensy base models on identical training data and then testing them on the same machine (a desktop computer) on identical test datasets consisting of 2 sets of 2000 Samples, with results averaged. Both models consistently made each prediction below the 50 ms requirement, with the smaller two-layer (64,32) LSTM model barely surpassing the Mean Average Error of the four-layer (128,128,128,128) model (5.03% vs 5.11%). When running the smaller model on the Teensy, the Mean Average Error was 5.44% (Fig. 14) for between 50 and 1000 ms into the future.

It is possible that the current method reaches an upper limit of precision at the current sample rate and input size, as long as the model is capable of predicting every cycle. Rendering the processing power of the Laptop unnecessary due to similar Mean Average Errors being achievable even on

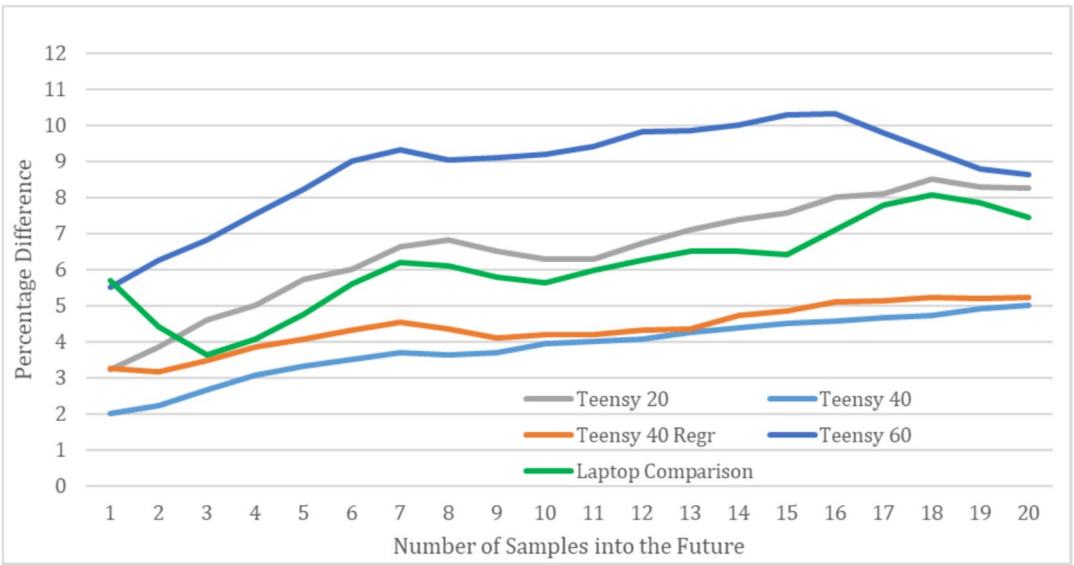

Fig. 12 Change in Percentage Difference over number of future predictions using Teensy 4.1. 1 Sample = 50 ms

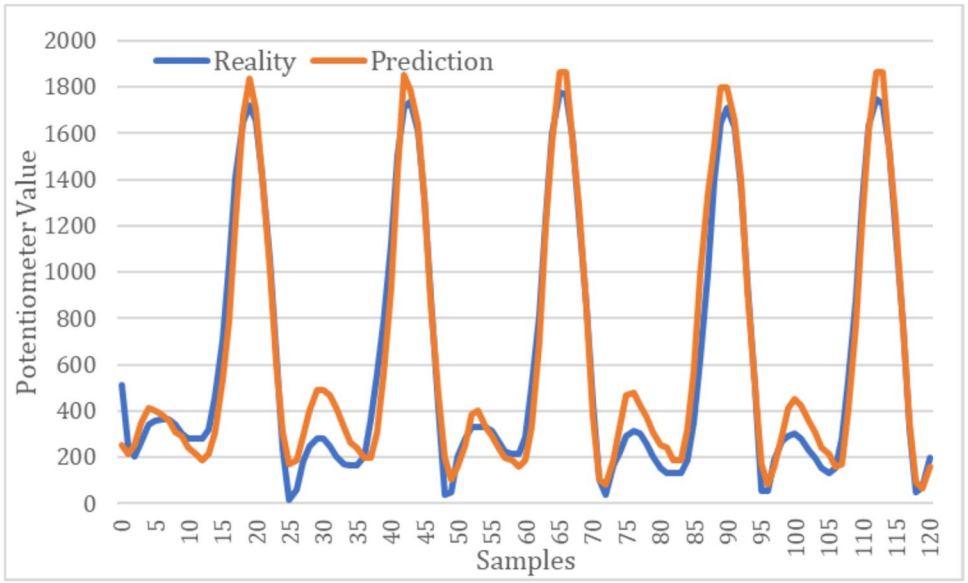

Fig. 13 *Teensy 4.1 Prediction vs Reality for* [8, 40] *input. Prediction made for 0.25 s into the future. 20 Hz Sampling Rate*

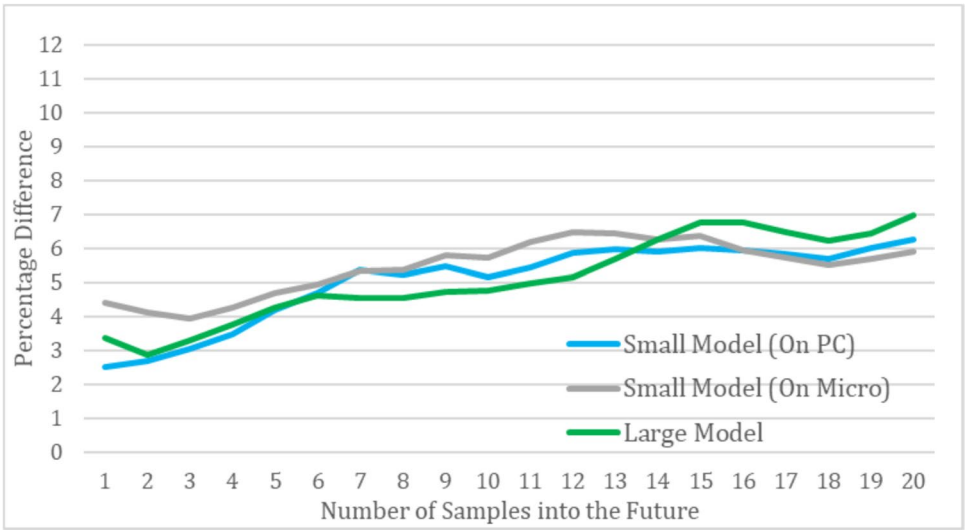

Fig. 14 Change in Percentage Difference over number of future predictions for Small Models on PC and Teensy Microcontroller, and Large Model on Laptop. 1 Sample = 50 ms

the microcontroller, with variance only due to differences in testing data in the original test. Therefore, this can be seen as the Microcontroller implementation being similarly "effective" as the Laptop implementation in predicting future joint angles, due to a similar Mean Average Error over the same prediction range of 50 ms to 1000 ms (1 to 20 samples).

5 Discussion & Conclusion

As was expected when comparing the Laptop Implementation to the ESP32 Microcontroller Implementation, the Laptop implementation benefits considerably from access to higher processing power by being able to run its model much faster. With both the HUZZAH32 ESP32 Feather possessing a 240 MHz dual core microcontroller and the Teensy 4.1 Possessing a 600 MHz dual core microcontroller, being not fairly comparable to the ROG Zephyrus M16 GU603ZX's 5 GHz, 14 Core CPU. Necessitating Practical Limitations to the Microcontroller implementation to be smaller in size and capacity to make knee angle predictions within reasonable timeframes (within the 50 ms program cycle time being considered ideal).

Despite this, when running both final implementation prediction systems on a 5,000-sample length dataset they had not been trained on, the average percentage differences between Reality and Prediction for both prediction systems were observed to be somewhat as expected (Fig. 15). With a decrease in accuracy (an increase in Mean Average Error) with a further prediction into the future.

For both the Microcontroller and Laptop Implementations, predictions had a clear linear increase in Mean

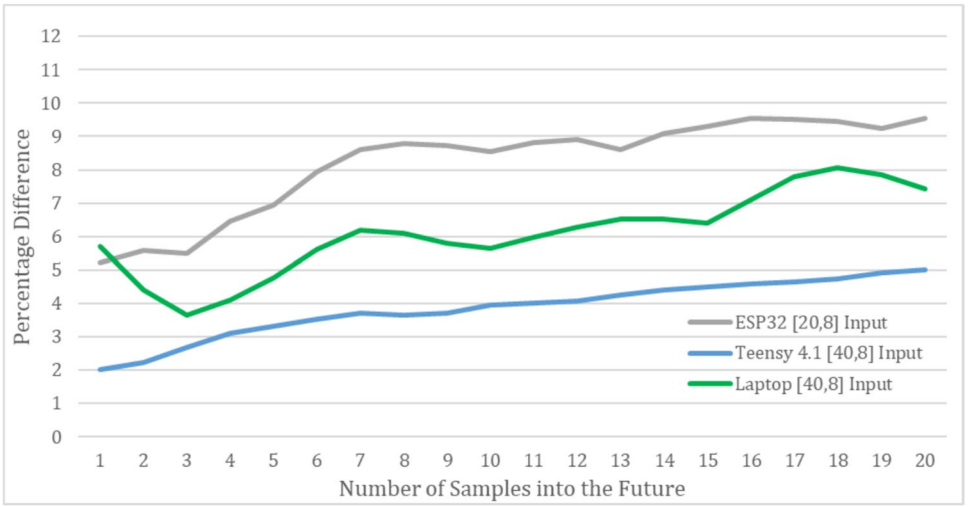

Fig. 15 Comparing Percentage Differences for the best Microcontroller, and Laptop Implementations. 1 Sample = 50 ms

Average Error for data predicted further into the future. Whilst the Laptop Implementation produced results of a higher overall accuracy than the ESP32, improvements to the model and usage of the Teensy 4.1 were enough to produce similar or superior results despite the considerable disparity in processing power. It is possible this may be due to reaching a limit in the current method's effectiveness where further processing power does not provide additional benefit.

In summary, there is some feasibility in implementing AI Prediction systems in Microcontrollers, although they are inflexible when presented with situations they have not been trained for or otherwise anomalous data. For a Low-Cost Implementation, while likely would not be suitable in situations where high precision is a necessity, they could have use as providing a general guide for the likely path of a user and is worthy of study as to such potential implementations within future work. In Effect, with this method a small and inexpensive onboard microcontroller is capable of achieving very accurate predictions that are similar or greater than that of more powerful devices, thereby removing the requirement of expensive tethered components.

As Observed from [39], p. 79]'s review of exoskeleton model implementations many models had a temporally small output window, with many looking 50–100 ms into the future at most, as opposed to this model's theoretical maximum of 1 s into the future. For example, [40] used IR Camera Data from 9 optical marks collected at 100 Hz and three EMG Electrodes collecting data at 2000 Hz spread across all joints, were applied as inputs to separate LSTM input layers before being concatenated, and fed to a fully connected network trained for 200 Epochs.

This system predicted only 50 ms into the future, but achieved accurate results, with RMSE Means of 0.464 Degrees, or an average percentage different of about 1.54% (across a ~ 30-degree variation). Similar papers such as [29], and [25] similarly involve knee-angle predicting LSTM's, achieving accuracies of 1.104 and 5.22 degrees for predicting 60 ms and 1 time-step respectively into the future.

These examples did not run upon microcontrollers, as such, when referring to examples of microcontroller-run models with referenced prediction accuracies, [23] used an STM32f407 and sampled IMU and EMG data at 100 Hz, using an RNN to predict up to 50 ms into the future with an error of ~ 2.93 degrees (~ 4% error). [20] meanwhile used a size 20 input of IMU data from the shank and foot, into a CNN model with two CNN layers, and was deployed onto an ESP32 Microcontroller. A Comparison is seen in Table 2.

In general, the exoskeleton control system produced here has similar results to that of other implementations, this model was designed to be capable of much higher depths of prediction. Over shorter prediction depths, the accuracy of the model was on average superior, for example, a prediction of 150 ms into the future as was used to control the motor resulted in a Mean Average Error of ~ 2.5%. This was achieved despite the far lower sampling rate of 20 Hz and computationally smaller model.

Table 2 Other examples of Knee Angle Tracking using Microcontroller-Loaded Models

Work	Input Type	Samp. Rate	Input Size	Processor	Model Type	Model Size	Output Size	Pred. Depth	MAE (est.)
This One	Pot/IMU	20 Hz	40	Teensy 4.1	LSTM	64,32	20	50–1000 ms	~ 5.44%
[22]	IMU/EMG	100 Hz	28/20	STM32	RNN	28/20, 32, 16	1	50 ms	~ 4%
[19]	IMU	Unk	20	ESP32	CNN	128, 64	10	20 ms to 200 ms	~ 3.75%

In effect, the model created within this paper both capable of predicting further into the future than many similar implementations, and when comparing within the prediction ranges of these models, achieved similar or superior Mean Average Error percentages.

The most prominent avenue for future work would be the combination of this prediction system with an actuator control system. Overcoming this current theoretical limitation would allow the system's functionality to be judged in physical implementation, and the difference between theoretical prediction and reality measured. This implementation is currently ongoing.

Author's Contributions Tom Slucock: Research and Manuscript Compilation. G.Howells, S.Hoque, and K.Sirlantzis: Supervisors, content advisory.

Funding This paper has received no external funding.

Data Availability Data available on Request.

Declarations

Ethics Approval The Author has read and approved this manuscript. Consent was given for the collection of Gait data in order to create predictive models.

Consent to Participate Ethics approval gained for collecting training data from participants. [CREAG012-11–22].

Consent for Publication The Author has agreed to publish this manuscript.

Conflicts of Interest The Author Declares no Conflicts of Interest.

References

1. Yagn, N.: Apparatus for Walking, Running, Jumping. United States of America Patent **420**, 179 (1890)
2. Vukobratovic, M., Hristic, D., Stojiljkovic, Z.: Development of active anthropomorphic exoskeletons. Med Biol Eng **12**(1), 66–80 (1974)
3. Specialty Materials Handling Products Operation General Electric Company, Hardiman I Prototype Project, 1968
4. T. Prendergast, "Healthcare expenditure, UK Health Accounts provisional estimates: 2022," Office for National Statistics, 2023
5. A. Jones, Interviewee, Ectron EksoNR cost. [Interview]. 21 September 2021
6. Shakti, D., Methew, L., Kumar, N., Kataria, C.: Effectiveness of robo-assisted lower limb rehabilitation for spastic patients: A systematic review. Biosens Bioelectron **117**, 403–415 (2018). https://doi.org/10.1016/j.bios.2018.06.027
7. Slucock, T.: A Systematic Review of Low-Cost Actuator Implementations for Lower-Limb Exoskeletons: a Technical and Financial Perspective. J Intell Robot Syst **106**(3), 1–31 (2022). https://doi.org/10.1007/s10846-022-01695-0
8. Zhou, Y., Sun, Z., Chen, B., Guang, G., Wu, X., Wang, T.: Human gait tracking for rehabilitation exoskeleton: adaptive fractional order sliding mode control approach. Intell Robot **3**(1), 95–112 (2023). https://doi.org/10.20517/ir.2023.05
9. Sherstinsky, A.: Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D **404**(132306), 28 (2020). https://doi.org/10.1016/j.physd.2019.132306
10. Orasan, I.L., Seiculescu, C., Caleanu, C.D.: A Brief Review of Deep Neural Network Implementations for ARM Cortex-M Processor. Electronics **11**(2545), 21 (2022). https://doi.org/10.3390/electronics11162545
11. Caldas, R., Mundt, M., Potthast, W., Neto, F.B.D.L.: A Systematic review of gait analysis methods based on inertial sensors and adaptive algorithms. Gait Posture **57**, 204–210 (2017). https://doi.org/10.1016/j.gaitpost.2017.06.019
12 Prasanth, H., Caban, M., Keller, U., Courtine, G., Ijspeert, A., Vallery, H., J. v. Zitzewitz,: Wearable Sensor-Based Real-Time Gait Detection: A Systematic Review. Sensors **21**(8), 2727–2755 (2021). https://doi.org/10.3390/s21082727
13. Goulermas, J.Y., Findlow, A.H., Nester, C.J., Liatsis, P., Zeng, X.-J., Kenney, L.P., Tresadern, P., Thies, S.B., Howard, D.: An Instance-Based Algorithm With Auxiliary Similiarity Information for the Estimation of Gait Kinematics From Wearable Sensors. IEEE Trans Neural Networks **19**(9), 1574–1582 (2008). https://doi.org/10.1109/TNN.2008.2000808
14. Sun, Y., Tang, Y., Zheng, J., Dong, D., Chen, X., Bai, L.: From sensing to control of lower limb exoskeleton: a systematic review. Annu Rev Control **53**, 83–96 (2022). https://doi.org/10.1016/j.arcontrol.2022.04.003
15. L. Rose, M. C. Bazzocchi and G. Nejat, "End-to-End Deep Reinforcement Learning for Exoskeleton Control," in International Conference on Systems, Man, and Cybernetics (SMC), Toronto, 2020. https://doi.org/10.1109/SMC42975.2020.9283306
16 Guo, Z., Wang, C., Song, C.: A Real-Time stable-control gait switching strategy for lower-limb rehabilitation. Plos One **15**(8), 19 (2020). https://doi.org/10.3389/fnins.2021.645374
17. Mundt, M., Thomsen, W., Witter, T., Koeppe, A., David, S., Bamer, F., Potthast, W., Markert, B.: Prediction of lower limb joint angles and moments during gait using artificial neural networks. Med Biol Eng Compu **58**(1), 211–225 (2020). https://doi.org/10.1007/s11517-019-02061-3
18. Sung, J., Han, S., Park, H., Cho, H.-M., Hwang, S., Park, J.W., Youn, I.: Prediction of Lower Extremity Multi-Joint Angles during Overground Walking by Using a Single IMU with a Low Frequency Based on an LSTM Recurrent Neural Network. Sensors **22**(53), 14 (2021). https://doi.org/10.3390/s22010053
19. Tan, J.-S., Tippaya, S., Binnie, T., Davey, P., Napier, K., Caneiro, J., Kent, P., Smith, A., O'Sullivan, P., Campbell, A.: Predicting Knee Joint Kinematics from Wearable Sensor Data in People with Knee Oseoarthritis and Clinical Considerations for Future

Machine Learning Models. Sensors **22**(446), 16 (2022). https://doi.org/10.3390/s22020446
20. Karakish, M., Fouz, M.A., Elsawaf, A.: Gait Trajectory Prediction on an Embedded Microcontroller Using Deep Learning. Sensors **22**(8441), 22 (2022). https://doi.org/10.3390/s22218441
21. Varma, V., Rao, R.Y., Vundavilli, P., Pandit, M., Budarapu, P.: A Machine Learning-Based Approach for the Design of Lower Limb Exoskeleton. Int J Comput Methods **9**(18), 22 (2022). https://doi.org/10.1142/S0219876221420123
22. Lee, T., Kim, I., Lee, S.-H.: Estimation of the Continuous Walking Angle of Knee and Angle (Talocrural Joint, Subtalar Joint) of a Lower-Limb Exoskeleton Robot Using a Neural Network. Sensors **21**(8), 2807 (2021). https://doi.org/10.3390/s21082807
23. Huang, Y., He, Z., Liu, Y., Yang, R., Zhang, X., Cheng, G., Yi, J., Ferreira, J.P., Liu, T.: Real-Time Intended Knee Joint Motion Prediction. IEEE Sens J **19**(23), 1558–1748 (2019). https://doi.org/10.1109/JSEN.2019.2933603
24. ST, "STM32F407G-DISC1," [Online]. Available: https://estore.st.com/en/stm32f407g-disc1-cpn.html. [Accessed 23 August 2023]
25. Z.-Q. Ling, G.-Z. Cao, Y.-P. Zhang, H.-R. Cheng, B.-B. He and S.-B. Cao, 2021 "Real-time Knee Joint Angle Estimation Based on Surface Electromyograph and Back Propagation Neural Network," in *18th International Conference on Ubiquitous Robots (UR)*, Gangneung-si, Gangwon-do, Korea https://doi.org/10.1109/UR52253.2021.9494639
26. Alemayoh, T.T., Lee, J.H., Okamoto, S.: Leg-Joint Angle Estimation from a Single Intertial Sensor Attached to Various Lower-Body Links during Walking Motion. Appl Sci **13**(4794), 1–17 (2023). https://doi.org/10.3390/app13084794
27. TensorFlow, "Get started with microcontrollers," [Online]. Available: https://www.tensorflow.org/lite/microcontrollers/get_started_low_level. [Accessed 23 Auguest 2023]
28. Tudor-Locke, C., Han, H., Aguiar, E.J., Barreira, T.V., Schuna, J.M., Jr., Kang, M., Rowe, D.A.: How fast is fast enough? Walking cadence (steps/min) as a practical estimate of intensity in adults: a narrative review. Br J Sports Med **52**(12), 776–788 (2017). https://doi.org/10.1136/bjsports-2017-097628
29. C. Zhu, Q. Liu, Q. Ai and S. Q. Xie, "An Attention-based CNN-LSTM Model with Limb Synergy for Gait Trajectory Prediction," in International Conference on Advanced Intelligent Mechatronics, Delft, Netherlands, 2021. https://doi.org/10.1109/AIM46487.2021.9517544
30. Hori, K., Mao, Y., Ono, Y., Ora, H., Hirobe, Y., Sawada, K., Inaba, A., Orimo, S., Miyake, Y.: Inertial Measurement Unit-Based Estimation of Foot Trajectory for Clinical Gait Analysis. Front. Physiol. **10**(1530), 12 (2020). https://doi.org/10.3389/fphys.2019.01530
31. Kuderle, A., Roth, N., Zlatanovic, J., Zrenner, M., Eskofier, B., Kluge, F.: The placement of Foot-Mounted IMU Sensors does affect the accuracy of spatial paramaters during regular walking. PLoS ONE **17**(6), 29 (2022). https://doi.org/10.1371/journal.pone.0269567
32. Pimoroni, "Adafruit HUZZAH32 – ESP32 Feather Board," [Online]. Available: https://shop.pimoroni.com/ products/adafruit-huzzah32-esp32-feather-board?variant=43873434122. [Accessed 10 August 2023]
33. Hosl, M., Schupfinger, A., Klich, L., Geest, L., Bauer, P., Bonfert, M.V., Afifi, F.K., Nader, S., Berweck, S.: Relationship between kinematic gait quality and caregiver-reported everday mobility in children and youth with spastic Cebebral Palsy. Eur J Paediatr Neurol **42**, 88–96 (2023). https://doi.org/10.1016/j.ejpn.2022.11.009
34 Kolaghassi, R., Al-Hares, M.K., Marcelli, G., Sirzlantzis, K.: Performance of Deep Learning Models in Forecasting Gait Trajectories of Children with Neurological Disorders. Sensors **22**(8), 18 (2022). https://doi.org/10.3390/s22082969
35. Asus, "ROG Zephyrus M16 (2022)," [Online]. Available: https://uk.store.asus.com/rog-zephyrus-m16-2022-203681799-90nr08r1-m000m0.html. [Accessed 10 August 2023]
36. TensorFlow, "tflite-micro," 08 May 2023. [Online]. Available: https://github.com/tensorflow/tflite-micro. [Accessed 09 May 2023].
37. Espressif, "tflite-micro-esp-examples," 07 May 2023. [Online]. Available: https://github.com/espressif/tflite-micro-esp-examples. [Accessed 09 May 2023]
38. Amazon, "Teensy 4.1 (Without Pins)," [Online]. Available: https://www.amazon.co.uk/Teensy-4-1-Without-Pins/dp/B088D3FWR7. [Accessed 10 August 2023]
39. Kolaghassi, R.: "Deep learning for Gait Prediction: An Application to Exoskeletons for Children with Neurological Disorders", University of Kent. Canterbury (2023). https://doi.org/10.3390/s22082969
40. L. J. Qingsong, W. Meng, Q. Liu and S. Q. Xie, "Individualized Gait Trajectory Prediction Based on Fusion LSTM Networks for Robotic Rehabilitation Training," in IEEE International Conference on Advanced Intelligent Mechatronics, Delft, Netherlands, 2021. 10.1109/ AIM46487.2021.9517616

Tom Slucock received the M.Eng degree in Electronic and Communications Engineering from the University of Kent,Canterbury, England in 2020. He worked with the University of Kent on a PhD to develop a low-cost assistive lower-limb exoskeleton making use of a microcontroller-based neural network prediction system, and has publishedacademically. He is now working within the avionics industry. His research interests include Robotics, Microcontrollers,Exoskeletons, and Neural Networks.

Gareth Howells is currently a Professor in the School of Computer Science and Electronic Engineering at the University of Essex. He has been involved in research relating to pattern recognition, Artificial Intelligence and assistive technologies for over 35 years and has published over 250 papers in the technical literature, co-editing two books and contributing to several other edited publications.

Sanaul Hoque received his B.Sc. degree in electrical and electronic engineering and the M.Sc. degree in computer engineering from the Bangladesh University of Engineering and Technology (BUET) and the Ph.D. degree in electronic engineering from the University of Kent, U.K., where he is currently a faculty member. He has authored more than 85 research articles . His research interests include assistive technologies, biometric security, image analysis and machine learning. He was in the editorial board of the*IET* Image Processing journal.

Konstantinos Sirlantzis is Professor of Applied Artificial Intelligence with the School of Engineering, Technology and Design, Canterbury Christ Church University (CCCU), Canterbury, Kent, U.K. Previously, he was Associate Professor of intelligent Systems at the School of Engineering, University of Kent, where he was the Head of the Robotics and Assistive Technologies Research Group and the Founding Director of Kent Assistive RObotics Laboratory (KAROL). He has a strong track record in artificial intelligence and neural networks for image analysis and understanding, robotic systems, with an emphasis in assistive technologies, and pattern recognition for biometrics-based security applications. He has authored over 150 peer-reviewed articles in journals and conferences. He has organized and chaired a range of international conferences and workshops.