Sensors and Materials, Vol. 33, No. # (2021) 1–##
MYU Tokyo

S & M ####

Implementation of Machine Learning and Deep Learning Algorithms with Dimensionality Reduction Methods for Internet of Things Gait Analysis and Monitoring Systems

Passara Chanchotisatien* and Chanvichet Vong

School of Information, Computer and Communication Technology,
Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani 12120, Thailand

(Received June 21, 2021; accepted August 24, 2021; online published December 6, 2021)

Keywords: HAFR, foot and ankle monitoring system, gyroscope, IMU, accelerometer, gait monitoring

In this paper, we present an end-to-end monitoring system, which is used for patients who have foot or ankle impairments. This system has been created to help orthopedic doctors optimize treatment for patients recovering from foot and ankle injuries. The system consists of three main parts: a wearable controlled ankle motion (CAM) boot equipped with inertial and load sensors, a web application that provides visual feedback obtained from sensors, and the implementation of machine learning and deep learning to analyze walking activity and gait. Sensors used on the CAM boot include an accelerometer, a gyroscope, and load cells. Values from sensors attached to the CAM boot are sent wirelessly to the database. The web application takes sensor values from the database and returns visual feedback on the patient's walking patterns in the form of different graphs. The graphs can be used to analyze and determine abnormalities in the patient's gait and serve as a visual aid for patients during rehabilitation. Sensor values obtained from the database are used to train machine learning and deep learning models to recognize and differentiate between seven activities performed by the patient. We study and compare three dimensionality reduction methods and six classifiers. As a result, we find that the joint incorporation of the dimensionality reduction method of sparse principal component analysis (PCA) and the classifier random forest (RF) gives the best result with an accuracy of 99.5%.

1. Introduction

Research on IoT-oriented gait analysis and activity recognition monitoring systems has increased due to the widespread usage of wearable smart devices and sensors.[1–4] This has enabled the collection of human activity data such as gait. Gait can be defined as the pattern observed from limb movements occurring during movement from one place to another. Various sensors such as inertial measurement unit (IMU) sensors and piezoelectric sensors can be used to observe, monitor, and distinguish between normal and abnormal gait, especially after sustaining injuries to the foot or ankle.[5–7] There have been many contributions in the form of

*Corresponding author: e-mail: passaraink@gmail.com
https://doi.org/10.18494/SAM.2021.3481

ISSN 0914-4935

smart shoes with attached sensors to analyze the gait of those with foot or ankle impairments.[8,9] For example, wireless joint ankle sensors can be used to measure joint rotations in three dimensions,[10] and pressure sensors can be attached to running shoes to measure the ground force applied by each foot.[11] There have been many attempts to classify, predict, and recognize various human activities by using various deep learning architectures and machine learning methods for both impaired and nonimpaired gait.[12–14]

Common causes of foot and ankle injuries include falls, vehicle accidents, and sports.[15,16] Such injuries are repeatedly experienced by many and could lead to temporary, long-term, or even permanent impairments depending on the severity of the patient's condition and how effectively the injury was treated. In cases where there are abnormalities in a patient's gait or walking pattern, it is important to identify these abnormalities and treat them in their earliest stages. Injuries that are not effectively treated could result in negative long-term effects or delayed recovery.

When consulting a doctor for foot and ankle injuries, patients are often given a cast or, in most cases, a controlled ankle motion (CAM) boot to wear and are instructed to walk in a certain way to ensure a quick recovery. For instance, a patient could be asked not to apply more than a certain weight on the injured foot or to fix certain irregularities in their gait cycle. Recommendations and instructions are often given to patients as auditory feedback. This process is mostly done through physical examination, which can often leave room for errors or failure to detect certain gait abnormalities or disorders as not all significant details can be captured through observation alone.[17] Additionally, auditory feedback can be difficult for patients to understand and visualize as they do not have visual feedback to assist them. Furthermore, doctors are unable to monitor the patient's progress after they have left the hospital. These problems could hinder the recovery of the patient and may even lead to permanent damage if the patient is unable to consistently follow recommendations or instructions given by the doctor.

In this study, we have built an end-to-end IoT system with the implementation of machine learning and deep learning algorithms to help combat and solve the problems mentioned above. The system will be used to analyze, monitor, and detect abnormalities in a person's gait. The system consists of three major components: the hardware design, the web application, and the implementation of machine learning and deep learning algorithms. The hardware design includes a CAM boot equipped with sensors that wirelessly transmit data values to the database. The web application retrieves sensor values from the database and outputs visual feedback of the patient's walking activity, allowing both patients and doctors to monitor the patient's gait in real-time through a web application. Data obtained from the sensors is used to train machine learning and deep learning models to determine the patient's gait or activity. We compare different classifiers and dimensionality reduction methods to find which give the best result.

The paper is structured as follows. Section 2 describes the system that was implemented. Section 3 explains gait analysis and shows graphs of different walking patterns from patients. Section 4 illustrates the machine learning and deep learning methodology. Section 5 discusses our experimental results. Finally, Sect. 6 presents our conclusions.

2. Proposed System

The proposed system gives insight into a patient's walking pattern and activity after an injury to the foot or ankle. The system can detect and differentiate between different types of gait patterns. The system consists of three main parts: the hardware, the web application, and the implementation of machine learning and deep learning algorithms. The hardware, which is the CAM boot, sends data wirelessly to the database. The web application outputs visual feedback of the patient's walking pattern using sensor values from the database. Finally, data collected in the database is used to train the machine learning and deep learning models. An overview of the system can be seen in Fig. 1.

2.1 Hardware implementation

The hardware consists of a CAM boot, an ESPino32 microcontroller, an IMU sensor, a load amplifier, and four load cells. The finished hardware can be seen in Fig. 2. The CAM boot is an adjustable device, which can be used with mild to severe leg, foot, and ankle injuries and is usually used to limit foot or ankle movement.[18] Consequently, a smaller difference in the

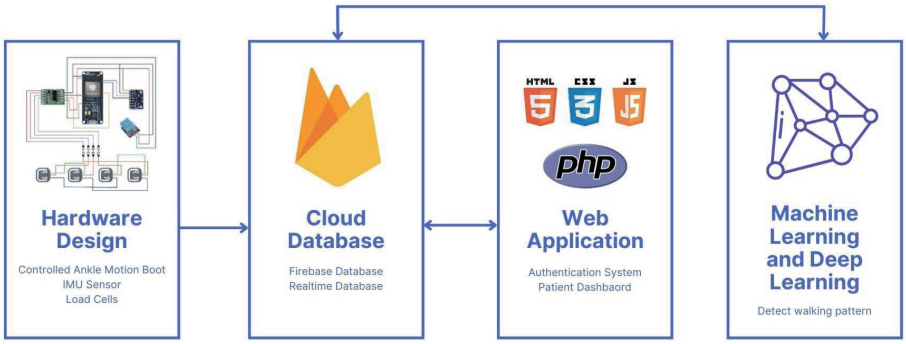


Fig. 1. (Color online) System overview.

Fig. 2. Hardware: IoT gait pattern monitoring boots.

orientation of the foot will be noticed while walking when wearing the boot. Here, the CAM boot was used as a platform to mount the sensors and collect the data. The sensors and microcontroller are described as follows.

The microcontroller used is the ESPino32 board produced by ThaiEasyElec. It is a low-cost and locally produced microcontroller that includes a Wi-Fi module and an ESP-WROOM-32 Bluetooth module. The Wi-Fi module has a range of 1 km, which allows the wireless transmission of sensor values to a database.

The IMU sensor contains both a gyroscope and an accelerometer. Here, a three-axis accelerometer and three-axis gyroscope are used. The IMU sensor used is the MPU6050 module, which employs an I2C interface. The accelerometer collects acceleration values of the injured foot, and the gyroscope collects angular velocity values. The IMU sensor can assist doctors in observing the patient's gait from patterns of the injured foot's angular velocity and acceleration. The raw data of the acceleration is configured to output values in the unit of *g* (the standard acceleration due to gravity). Then, the acceleration values are multiplied by 9.8 to obtain acceleration in m/s^2. On the other hand, angular velocity values are given in the unit of degrees per second.

The load cells and load amplifier are used to help monitor the weight exerted by the patient on the injured foot. Four load cells are used, which are attached to the insole of the CAM boot. The four load cells are connected to an HX711 load amplifier. The load amplifier combines four load sensors into a standard four-wire Wheatstone bridge configuration, which outputs the total weight value from all four sensors combined when weight is applied. Finally, the circuit is powered by a power bank attached to the CAM boot.

2.2 Web application implementation

Patients and doctors can log into the web application to track and monitor the progress of recovery and view the walking activity of the patient's injured foot. After data are sent wirelessly to the database from the CAM boot, the data are displayed on the dashboard of the web application for patients and doctors to see. Information displayed includes acceleration graphs, load graphs, and angular velocity graphs, which can help give insight on the patient's gait such as the speed at which the patient is walking, how much weight is exerted, and the number of steps per second. The front end of the web application was created using HTML and CSS. The back end was created with PHP, and a Firebase real-time database was used, which is a cloud-hosted NoSQL database. With Firebase, data can be stored and synced between users in real-time. The user interface of the web application is shown in Fig. 3.

3. Gait Analysis

In this section, we compare and observe different types of gait. We tested our system on three different patients in three different scenarios: walking heel first, walking foot flat, and walking toe first. Each scenario will be analyzed in detail in the following sections using the dashboard from the web application we have created. The different types of gait will be described on the

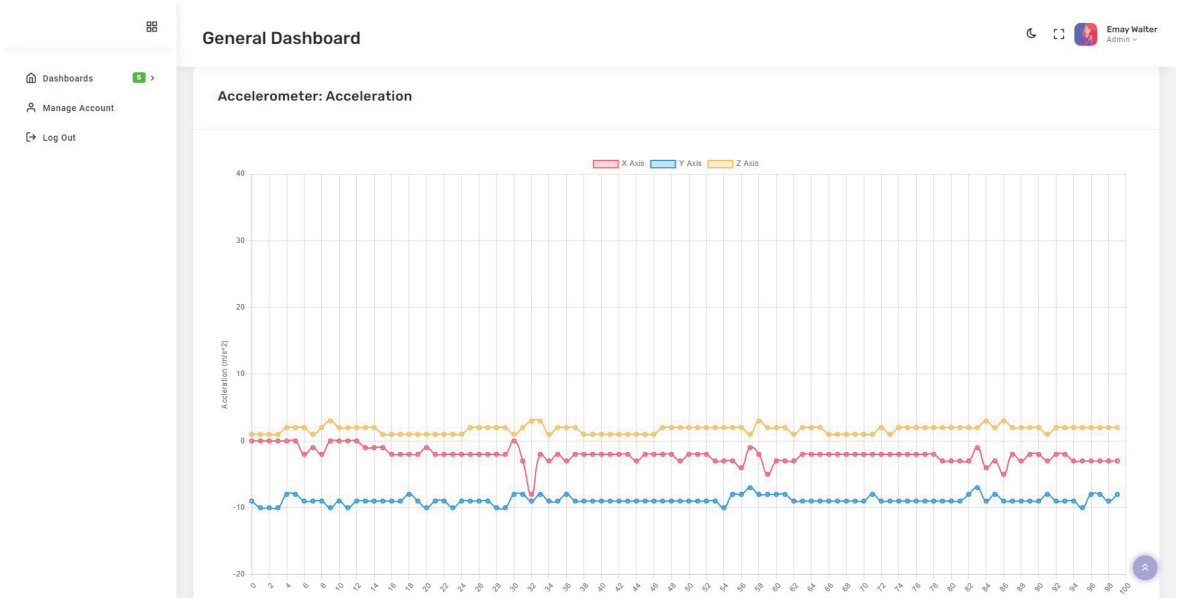


Fig. 3. (Color online) User interface of dashboard inside the web application.

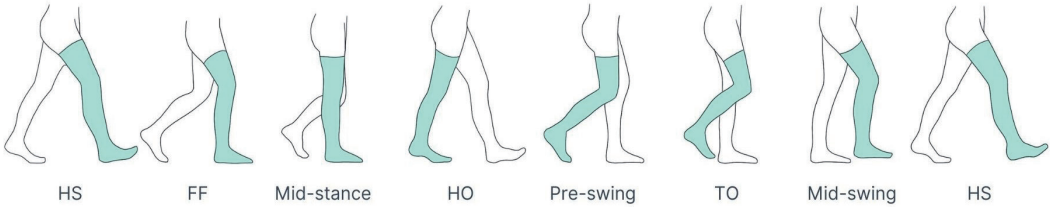


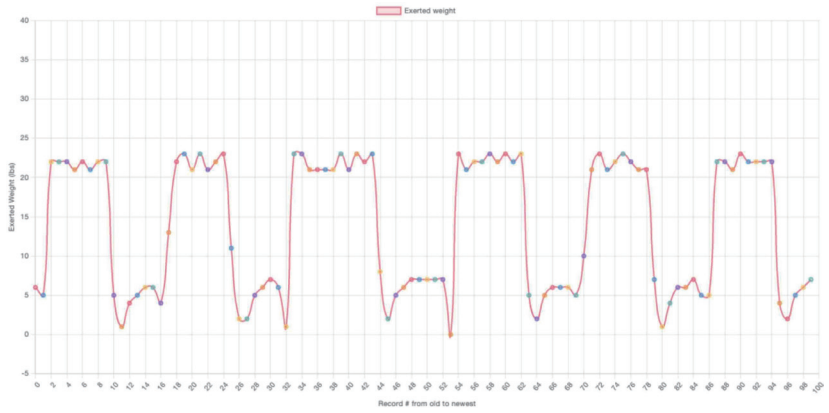
Fig. 4. (Color online) Normal gait cycle.

basis of patterns of exerted weight, acceleration, and angular velocity in the *x*, *y*, and *z* axes. The scenario of walking heel first is that of a normal gait, while the other two scenarios are instances of abnormal gait. When walking heel first, the participant strikes their heel to the ground first at the beginning of each gait cycle. To first understand and recognize abnormal gait, we study the normal gait cycle, which is shown in Fig. 4. There are four commonly observed events in the gait cycle: heel strike (HS), foot flat (FF), heel-off (HO), and toe-off (TO).[19,20] HS is when the heel strikes the ground, FF is when the foot is placed flat on the ground, during which the most weight is applied, HO is when the heel leaves the ground, and TO is when the toe leaves the ground.

3.1 Scenario 1: Walking heel first

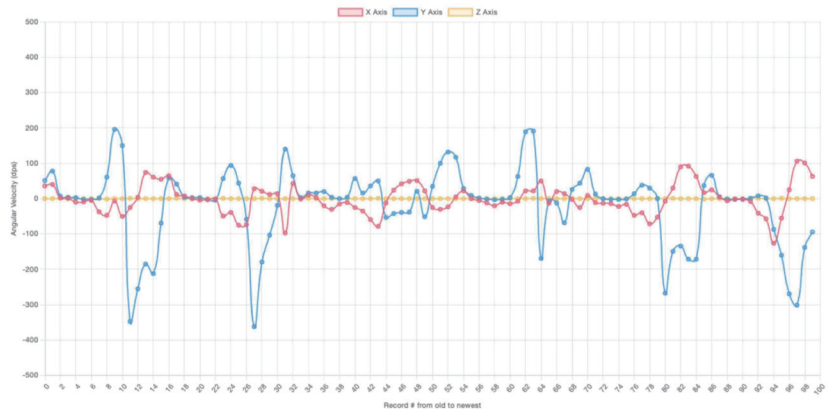
Walking heel first is considered a normal gait. Walking heel first involves striking the heel on the ground first at the beginning of each gait cycle. The line graphs for exerted weight, acceleration, and angular velocity are shown in Figs. 5(a)–5(c), respectively. Each graph displays a total of 100 records, corresponding to a walking time of 5 s (oldest record to newest record).

From these graphs, we can analyze the participant's gait in terms of cadence, walking speed, and how much weight the patient exerts on the injured foot. From the weight graph, we can see that the patient takes 1.2 steps per second and exerts a weight of around 10 kg in each step. We can spot the four main stages of the gait cycle clearly in the weight graph. Each stage is highlighted in Fig. 6.



(a)

(b)



(c)

Fig. 5. (Color online) Graphs of (a) exerted weight, (b) acceleration, and (c) angular velocity for walking heel first.

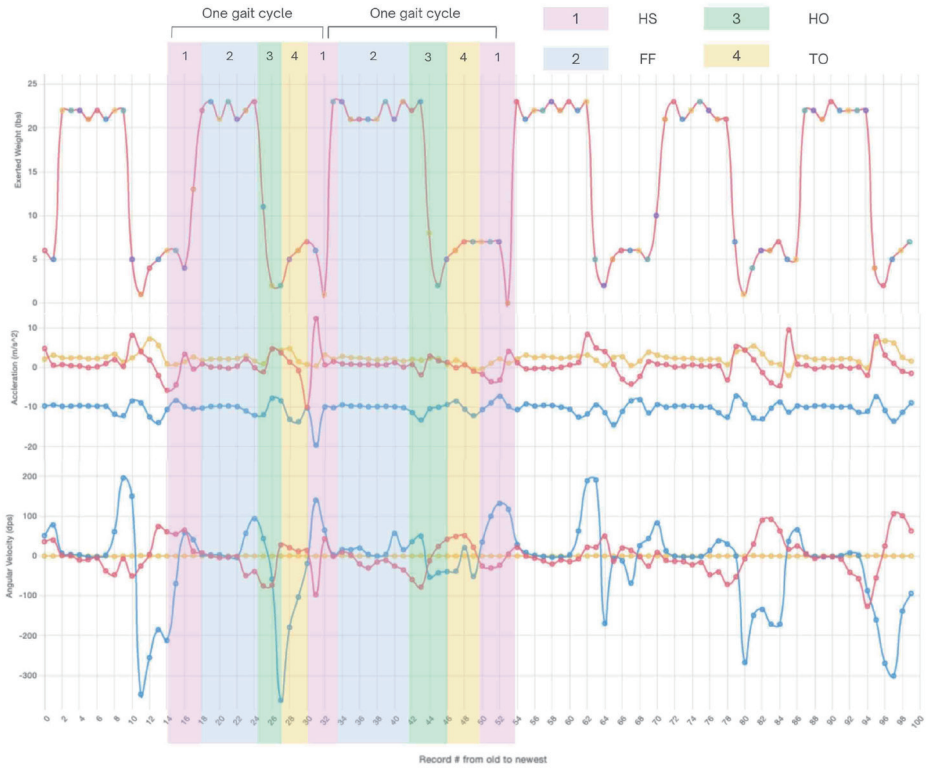


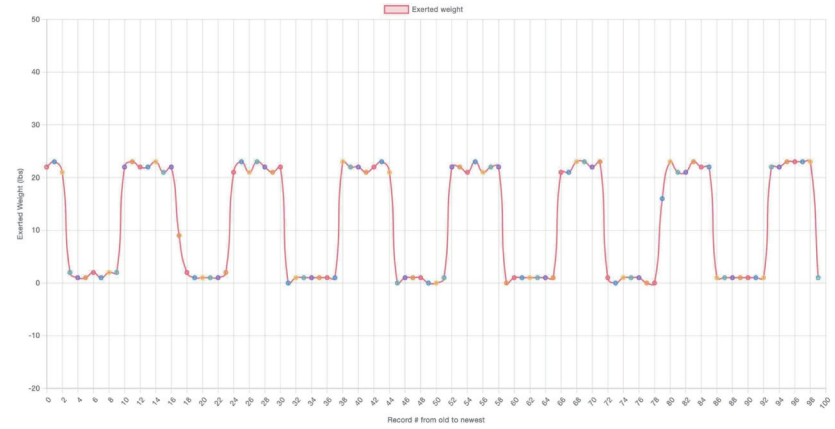
Fig. 6. (Color online) Detected gait events labeled on graphs for walking heel first.

Following our gait cycle diagram in Fig. 4, we see that the gait cycle begins with the striking of one's foot on the ground. We can spot this stage (HS) by a small increase in the exerted weight graph (marked by number 1 in Fig. 6) before the weight reaches its peak. This is due to the increased pressure on the heel but not yet the whole foot. The next stage detected after the heel strike in the graph is the FF (number 2 in Fig. 6). In this stage, the most weight is applied on the foot, which explains the significant increase in the exerted weight as can be seen from the graph. After that, the participant lifts the heel off from the ground to take the next step. This decreases the weight on the foot as we can see from the drop in the graph of the exerted weight (number 3). In the last stage, the toe leaves the ground (TO) to prepare for the next HS. We can also detect these gait events in the acceleration and angular velocity graphs. It is noticeable that during the FF stage, there is no or very little change in the acceleration and angular velocity in all axes. It is also important to point out that the angular velocity values in the z axis are always near zero as the orientation in this direction is limited due to the CAM boot. This trend can be observed in walking FF, walking toe first, and walking heel first.

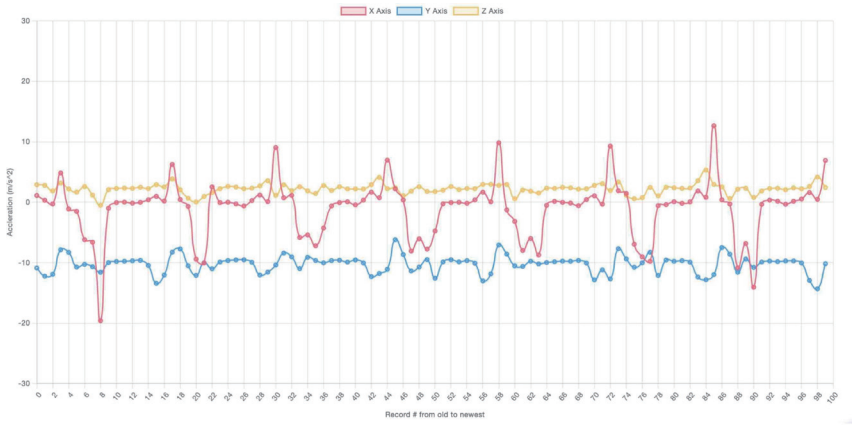
3.2 Scenario 2: Walking FF

In the second scenario, the participant walks FF. This means that the participant does not strike their heel on the ground first and transit to the FF stage, but instead, they instantly place

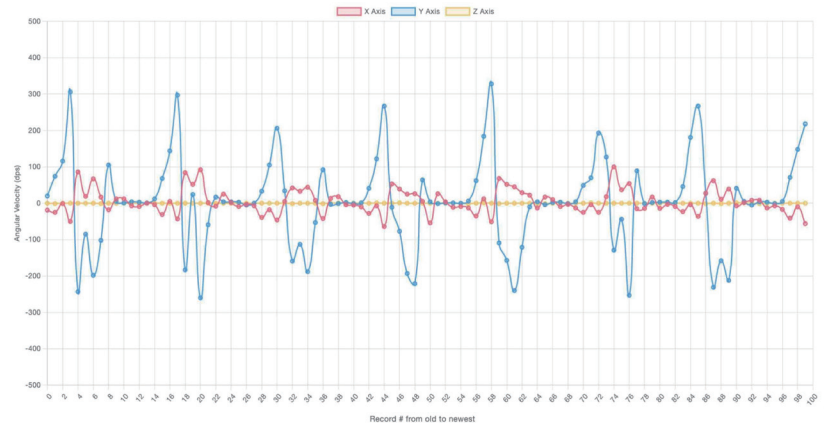
their foot flat on the ground. The exerted weight graph, acceleration graph, and angular velocity graph for FF are shown in Figs. 7(a)–7(c), respectively.



(a)



(b)



(c)

Fig. 7. (Color online) Graphs of (a) exerted weight, (b) acceleration, and (c) angular velocity for walking FF.

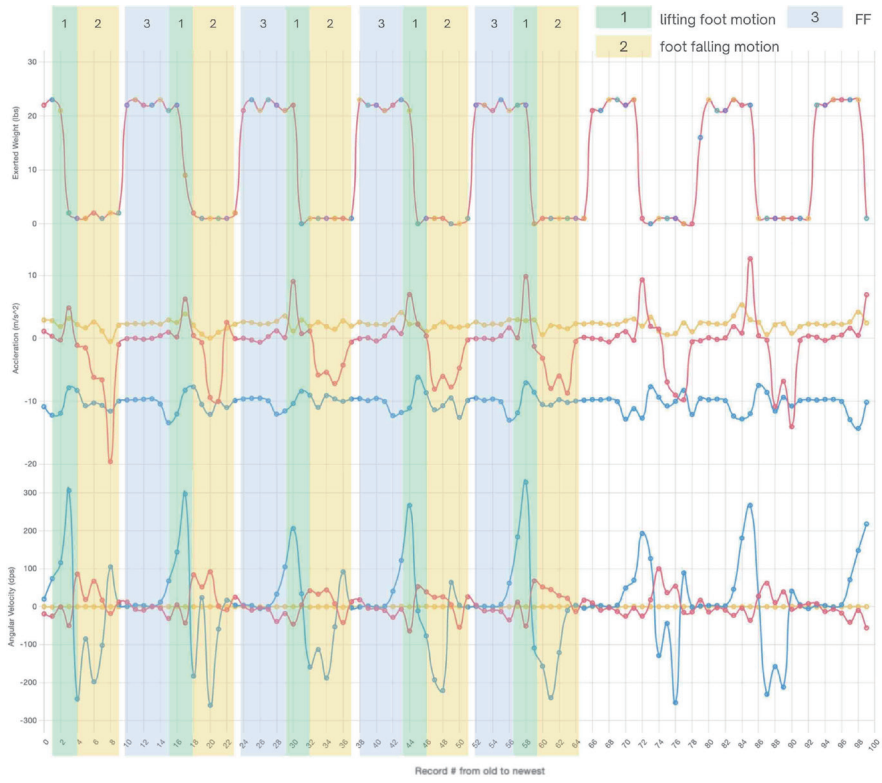


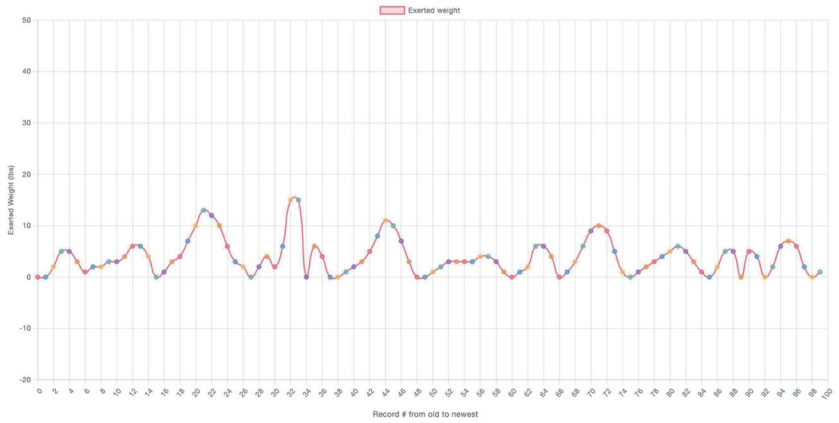
Fig. 8. (Color online) Detected gait events labeled on graphs for walking foot flat.

From Fig. 8, we see three main detectable gait events: lifting foot motion (marked by number 1), foot falling motion (number 2), and FF (number 3). Unlike the load cell graph for walking heel first, we see that there is no gradient change in the exerted weight as there is no HS motion in walking FF. The participant skips the stage HS, which is the stage where the heel cushions some of the weight, and instantly places their foot flat on the ground. As a result, at each step, the increase in exerted weight is abrupt, with the value increasing from almost zero to 10 kg immediately.

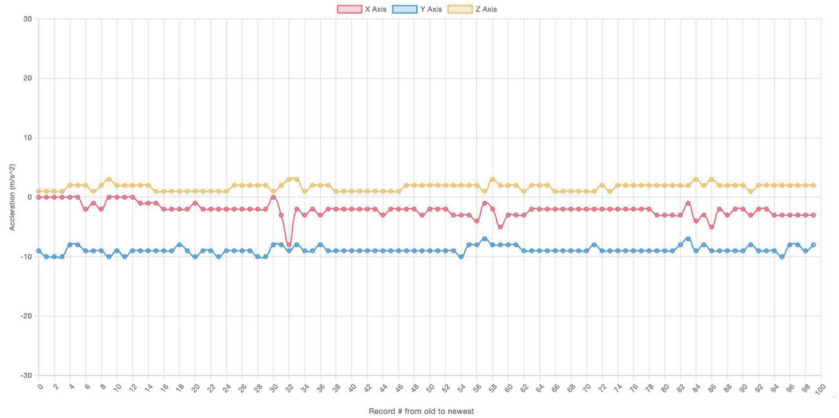
From the acceleration and angular velocity graphs, we see that there is little or no change in acceleration or angular velocity during the FF stage, which was also the case in the previous scenario. In addition, it can be seen that a significant positive increase in acceleration and angular velocity values occurs when the participant performs the lifting foot motion. Similarly, there is a significant negative increase in acceleration and angular velocity values when the participant performs the foot falling motion.

3.3 Scenario 3: Walking toe first

In this scenario, the participant walks toe first, where instead of striking the heel onto the ground first at the beginning of each gait cycle, the toe is used instead. Participants with this gait only exert a small amount of weight on the toes of the foot of interest and nowhere else. The exerted weight, acceleration, and angular velocity graphs are shown in Figs. 9(a)–9(c),



(a)



(b)

(c)

Fig. 9. (Color online) Graphs of (a) exerted weight, (b) acceleration, and (c) angular velocity for walking toe first.

respectively. We can see that the steps taken are more random and that little force is applied at each step. We also notice there is little change in the acceleration and angular velocity values.

4. Machine Learning and Deep Learning Implementation Methodology

We compare multiple machine learning algorithms using dimensionality reduction methods and deep learning neural networks. The diagram in Fig. 10 illustrates our methodology to investigate and compare different algorithms. We first load the data signals, which consist of the load, acceleration, and angular velocity in the *x*, *y*, and *z* axes. Then, we segment the data into frames of 4 s and encode labels in the segmented data.

To compare different machine learning classifiers and dimensionality reduction methods, we extract features from our data. A pipeline is constructed containing a standard scaler, a dimensionality reduction method, and a classifier. It is used to prevent data from leaking between training and test data points. The data that pass through the pipeline are cross-validated via stratified K-fold cross-validation. For deep learning, data are split into training and test datasets. The data are then scaled and trained through the deep learning architecture. Further information on each stage will be given in the following subsections.

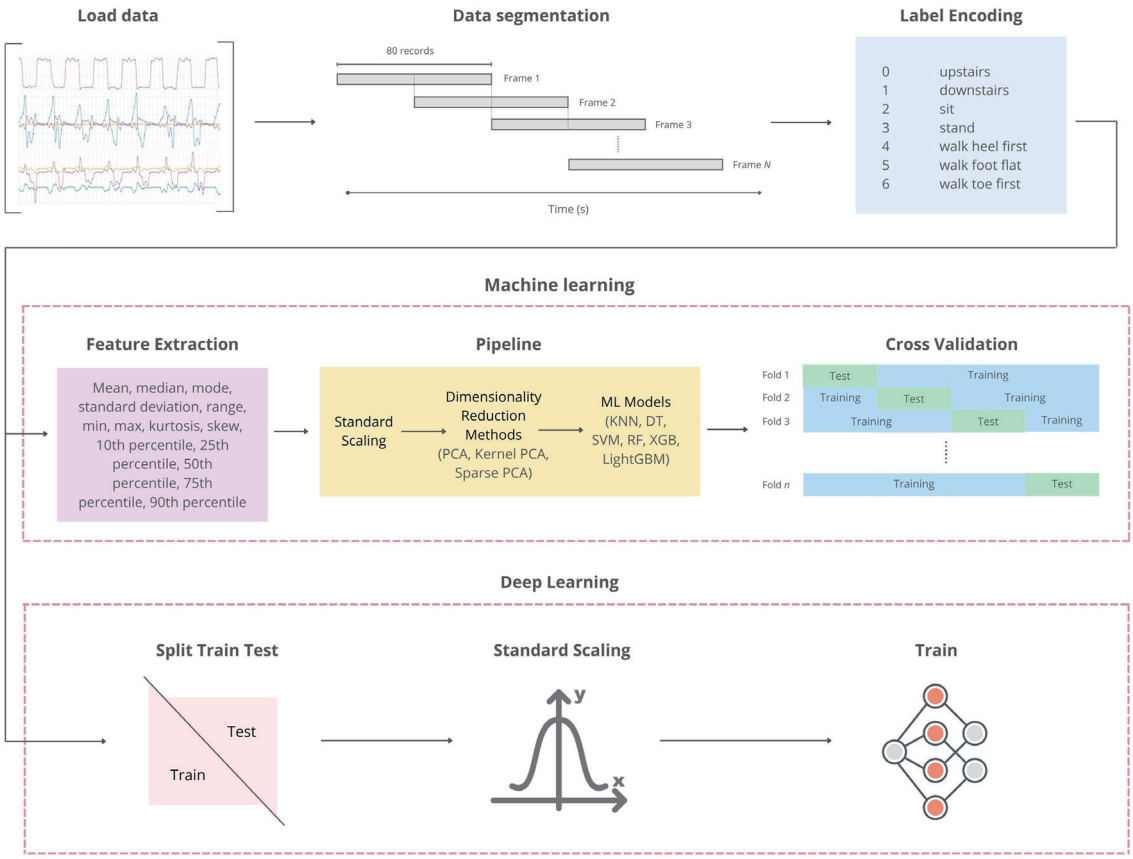


Fig. 10. (Color online) Machine learning and deep learning methodology.

4.1 Data collection

Our dataset consists of six people, referred to as participants A–F. Details of each participant are listed in Table 1. The data recorded comprise *x*-axis, *y*-axis, and *z*-axis acceleration signals, *x*-axis, *y*-axis, and *z*-axis angular velocity signals, and load (exerted weight on foot). The collected data are stored as JSON files.

We collect the data in seven scenarios: sitting position, standing position, walking upstairs, walking downstairs, walking foot first, walking toe first, and walking heel first. These activities have been suggested by a doctor to be the basic activities carried out by a person in a day. Table 2 shows the number of records collected for each activity. Each participant in the dataset is instructed to perform each activity repeatedly for an amount of time.

4.2 Data segmentation

The results of a previous study conducted to determine a suitable sampling rate suggested a sampling rate of 22 Hz.[21] As a result, our data are collected at a frequency of 22 Hz. The data are segmented into frames of 4 s each with an overlapping rate of 50%, following suggestions regarding the window size and sliding window techniques made in Refs. 22 and 23. Therefore, the frame size is equal to 88 records. One frame will only consist of records belonging to one activity. We iterate over each JSON data file to create frames. Table 3 shows the number of frames extracted for each activity.

Table 1
Details of each participant in the dataset.

Participant	Age	Gender	Weight (kg)	Height (cm)
A	17	Female	44	158
B	19	Female	54	160
C	53	Female	51	155
D	23	Male	83	177
E	24	Male	80	172
F	57	Male	78	172

Table 2
Number of records extracted for each activity.

Activity	Number of records
Sitting	27603
Standing	27606
Walking upstairs	19800
Walking downstairs	19800
Walking foot flat	34407
Walking toe first	27616
Walking heel first	27630

Table 3
Number of frames collected for each activity.

Activity	Number of frames
Sitting	612
Standing	612
Walking upstairs	449
Walking downstairs	448
Walking foot flat	764
Walking toe first	612
Walking heel first	612

4.3 Feature extraction

We extract 14 features from each of our seven signals of load, *x*-axis acceleration, *y*-axis acceleration, *z*-axis acceleration, *x*-axis angular velocity, *y*-axis angular velocity, and *z*-axis angular velocity. The features are mean, median, mode, standard deviation, max, min, range, skew, kurtosis, 10th percentile, 25th percentile, 50th percentile, 75th percentile, and 90th percentile. The equations for the standard deviation, range, skew, kurtosis, and percentile are shown in Eqs. (1)–(5), respectively.

$$\sqrt{\frac{\Sigma\left(x_i-\mu\right)^2}{N}} \tag{1}$$

$$\max(X) - \min(X) \tag{2}$$

$$g_1 = m_3 / m_2^{3/2}, \text{ where } m_i = \frac{1}{N}\sum_{n=1}^{N}\left(x_n - \bar{x}\right)^i \text{ and } \bar{x} \text{ is the sample mean} \tag{3}$$

$$kurt = \mu^4/\sigma^4, \text{ where } \mu^4 \text{ is the fourth central moment and } \sigma \text{ is the standard deviation} \tag{4}$$

$$n = (P/100) \times N, \text{ where } P \text{ is the percentile and } n \text{ is the ordinal rank of a given value} \tag{5}$$

4.4 Data scaling

StandardScaler is used to reduce large differences between points. This helps lessen the impact on the model training. Not scaling the data may lead to skewed results when training the model. StandardScaler transforms the data to make the mean of the distribution of each feature equal to 0 and the standard deviation equal to 1. We use the sci-kit learn library to scale our data.

4.5 Dimensionality reduction approaches

When training with a large number of features, models may be negatively affected by the curse of dimensionality and poor generalization.[24] Dimensionality reduction approaches have been shown to improve the performance of machine learning models.[25] In this study, three dimensionality reduction approaches are used: principal component analysis (PCA), Kernel PCA (KPCA), and Sparse PCA.

PCA is a dimensionality reduction method that converts a set of features from a higher dimension into a lower dimension. It achieves this by maximizing the variance in the new subspace by selecting the fittest principal components.[26] PCA is a linear algorithm. Kernel PCA extends PCA by applying a kernel to the dataset to improve the performance for features that are not linearly separable.[27] Sparse PCA extends PCA differently. PCA uses linear combinations of all of the original features, producing new features that are difficult to interpret. Sparse PCA solves this problem by using linear combinations of a few features.[28] The libraries used to perform dimensionality reduction are listed in Table 4.

Table 4
Dimensionality reduction methods and their libraries.

Dimensionality reduction method	Library
PCA	sklearn.decomposition.PCA
Kernel PCA	sklearn.decomposition.KernelPCA
Sparse PCA	sklearn.decomposition.SparsePCA

Table 5
Classifiers and their libraries.

Classifier	Library
KNN	sklearn.neighbors.KNeighborsClassifier
DT	sklearn.tree.DecisionTreeClassifier
SVM	sklearn.svm.SVC
RF	sklearn.ensemble.RandomForestClassifier
XGB	xgboost.XGBClassifier
LightGBM	lightgbm.LGBMClassifier

4.6 Machine learning classification using cross-validation

Various classifiers have been used and tested in previous studies.[29,30] In Ref. 31, it was shown that ensemble methods can improve the classification performance. Therefore, we compare both simple and ensemble classifiers in this study. We test the following machine learning classifiers on our dataset: K-nearest neighbor (KNN), decision tree (DT), support vector machine (SVM), random forest (RF), XGBoost (XGB), and LightGBM. To evaluate and compare each classifier, we apply Stratified K-fold cross-validation from the sci-kit learn library, where $K = 10$. The libraries used for the learning classifiers are listed in Table 5.

4.7 Deep learning architecture

Recently, there have been many approaches where neural networks are used to classify and differentiate between gait or human activities.[32,33] In this study, we also implement a simple convolutional neural network for evaluation and comparison. It consists of 2D convolution layers, a dropout layer, and dense layers. All layers use the ReLU activation function except for the last layer, which uses SoftMax. The Adam optimizer is utilized with the default parameters. The deep learning architecture is shown in Fig. 11.

5. Experimental Results and Discussion

Previously, we described the process by which we investigate and compare six machine learning classifiers (KNN, DT, SVM, RF, XGB, and LightGBM) and a simple convolutional neural network. Several dimensionality reduction approaches are applied to each machine learning classifier to evaluate and determine the best combination. The default hyperparameters are used for a fair comparison. The default hyperparameters of each classifier are listed in Table 6 for reference. The performance results are measured on the basis of accuracy, F1, precision, and recall scores.

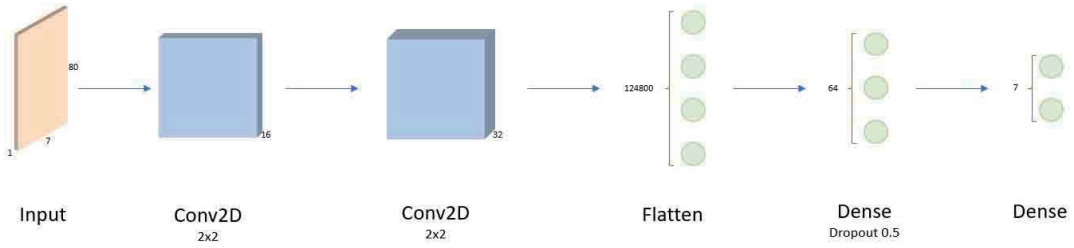


Fig. 11. (Color online) Deep learning architecture.

Table 6
Classifiers and their default hyperparameters.

Classifier	Default hyperparameters
KNN	algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=-1, n_neighbors=5, p=2, weights='uniform'
DT	ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort='deprecated', random_state=10, splitter='best'
SVM	C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf', max_iter=-1, probability=False, random_state=10, shrinking=True, tol=0.001, verbose=False
RF	bootstrap=True, ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1, oob_score=False, random_state=10, verbose=0, warm_start=False
XGB	base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=3, min_child_weight=1, missing=None, n_estimators=100, n_jobs=1, nthread=None, objective='binary:logistic', random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,seed=None, silent=None, subsample=1, verbosity=1
LightGBM	boosting_type='gbdt', class_weight=None, colsample_bytree=1.0, importance_type='split', learning_rate=0.1, max_depth=-1, min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=31, objective=None, random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=1.0, subsample_for_bin=200000, subsample_freq=0

5.1 Results of machine learning classifiers with dimensionality reduction methods

We use stratified K-fold cross-validation with $K = 10$ and shuffle = true to evaluate the performance of our models. This evaluation method splits a dataset into K sets, where each set is trained $K - 1$ times and tested once. Each set conserves the same ratio of samples of each class. The stratified K-fold process can be seen in Fig. 10. In Table 7, we show the classification performance (accuracy, precision, recall, and F1 scores) of the six classifiers (KNN, DT, SVM,

Table 7
Classifiers and their scores with dimensionality reduction (in %).

Dimensionality reduction	Classifier	Accuracy (%)	F1 (%)	Precision (%)	Recall (%)
PCA	KNN	96.37	96.41	96.73	96.17
	DT	91.36	90.76	91.30	90.47
	SVM	97.90	97.98	98.11	97.88
	RF	97.68	97.65	97.91	97.47
	XGB	97.68	97.64	97.85	97.49
	LightGBM	97.90	97.88	98.16	97.68
Kernel PCA	KNN	96.37	96.41	96.73	96.17
	DT	91.67	91.15	91.64	90.89
	SVM	97.90	97.98	98.11	97.88
	RF	97.66	97.61	97.98	97.35
	XGB	97.98	97.91	98.08	97.78
	LightGBM	98.15	98.13	98.31	97.99
Sparse PCA	KNN	95.95	95.97	96.30	95.74
	DT	96.08	95.75	96.07	95.54
	SVM	97.90	97.97	98.12	97.87
	RF	**99.53**	**99.54**	**99.59**	**99.51**
	XGB	**99.29**	**99.29**	**99.34**	**99.25**
	LightGBM	**99.41**	**99.42**	**99.45**	**99.41**

RF, XGB, and LightGBM) with the incorporation of dimensionality reduction methods (PCA, Kernel PCA, and Sparse PCA). The top three accuracy, F1, precision, and recall scores are in bold. From the table, it is apparent that the RF classifier with Sparse PCA incorporated gives the highest accuracy score of 99.53%. Additionally, we can observe that RF, XGBoost, and LightGBM generally outperform the other three classifiers. The main reason for this is that RF, XGBoost, and LightGBM are ensemble classifiers, which generally perform better than simple classifiers. RF creates multiple decision trees and combines their results to obtain a better result. XGBoost and LightGBM use gradient boosting. Here, trees are continuously added to the initially built tree to cover the weakness of the previous model until it reaches the set amount specified in the hyperparameter called n_estimators. This is why ensemble methods generally have superior performance. We also notice that PCA gives the lowest scores when incorporated with other machine learning classifiers, while Sparse PCA significantly improves the performance of the classifiers, with the three highest performances obtained with Sparse PCA. Since we know that Sparse PCA utilizes linear combinations of only a selected set of features from the dataset, this could mean that there are features that are unnecessary or have a negative impact on the dataset. Fig. 12 shows the importance of each feature decided from the combination of RF and Sparse PCA from highest importance to lowest importance. We can see that many features are considered unnecessary.

5.2 Deep learning results

The deep learning model is constructed using TensorFlow 2 following the architecture in Fig. 11. For deep learning, data are split into training and test datasets with 80% being the train dataset and 20% being the test dataset. A small model is chosen to minimize the running time

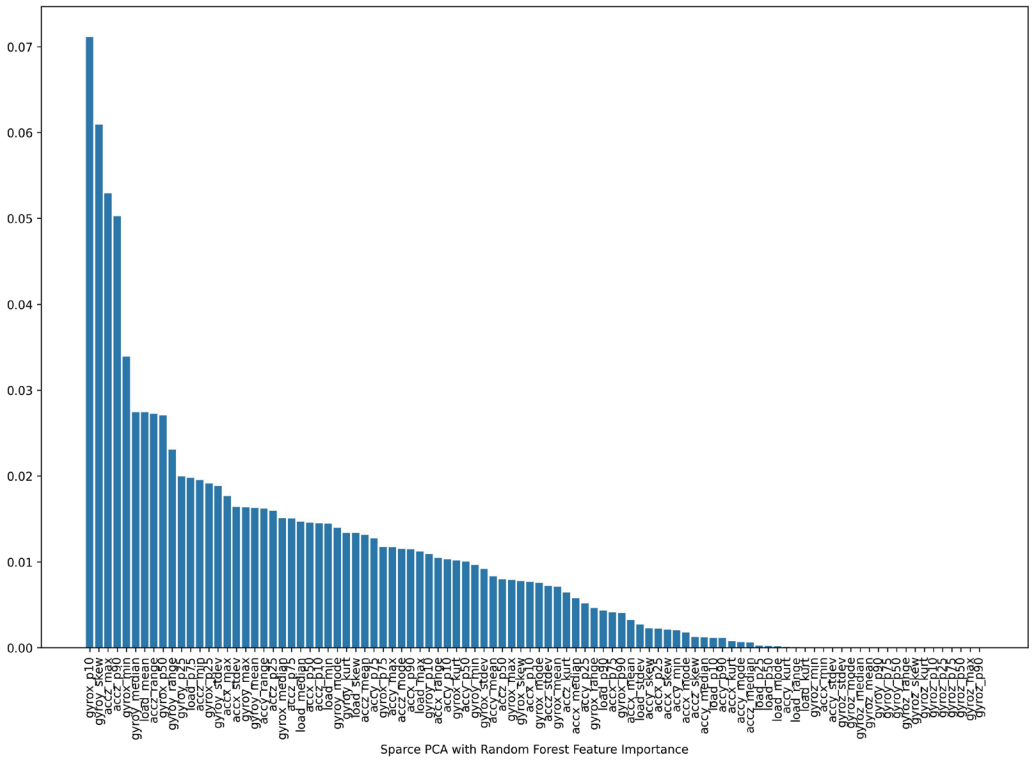


Fig. 12. (Color online) Ranked importance of features from sparse PCA and RF.

while still maintaining high performance. The accuracy, F1, precision, and recall scores are 96.71, 96.62, 96.90, and 96.44%, respectively.

5.3 Comparison with previous studies

In this section, we compare the results obtained in this study with those in the literature. Similar to in previous studies, we showed that RF outperforms other classifiers. In Ref. 34, in which the dimensionality reduction methods of PCA and linear discriminant analysis were compared, the highest accuracy of 87.5% was obtained using RF and PCA, with acceleration in the *x*, *y*, and *z* axes used as the only features. In Ref. 35, different classifiers were tested against three different datasets. The highest accuracy scores were again obtained from RF and ranged from 96.74 to 100%. In Ref. 36, CNN and LSTM were applied to the UCI HAR dataset and the highest accuracy obtained was 92.13%. It is important to note that different datasets were used in each study and that the results obtained in this paper are based on an original dataset collected specifically for this study.

6. Conclusions

We have created a system that can be used to monitor and analyze the foot pressure and movement of a person with foot or ankle impairments. The system will help orthopedic doctors

optimize treatments for patients. The system consists of a CAM boot with attached sensors, a web application, and the implementation of machine learning and deep learning algorithms, which help determine and differentiate between normal and abnormal gait. We have investigated and compared the classification performance of machine learning classifiers with incorporated dimensionality reduction methods. Our experimental results have proven that the RF classifier with Sparse PCA incorporated outperforms all other methods. As future work, we would like to explore more complex deep learning architectures and expand our activity dataset.

References

1 W. Tao, T. Liu, R. Zheng, and H. Feng: Sensors **12** (2012) 2255. https://doi.org/10.3390/s120202255
2 Y. Huang, Y. Chen, H. Wu, and Y. Huang: Sens. Mater. **31** (2019) 629. https://doi.org/10.18494/SAM.2019.2167
3 F. Sun, C. Mao, X. Fan, and Y. Li: IEEE Internet Things J. **6** (2019) 820. https://doi.org/10.1109/JIOT.2018.2860592
4 A. Muro-de-la-Herran, B. Garcia-Zapirin, and A. Mendez-Zorilla: Sensors **14** (2014) 3362. https://doi.org/10.3390/s140203362
5 T. Yu and C. Wu: Proc. 2019 Int. Conf. Machine Learning and Cybernetics (ICMLC, 2019) 1–5. https://doi.org/10.1109/ICMLC48188.2019.8949256
6 Y. Igarashi, W. Hayasaka, K. Tsurumiya, H. Tsukamoto, T. Suda, T. Iwami, and Y. Shimada: Proc. 2020 IEEE 2nd Global Conf. Life Sciences and Technologies (LifeTech, 2020) 23–27. https://doi.org/10.1109/LifeTech48969.2020.1570619170
7 A. E. Minarno, W. A. Kusuma, H. Wibowo, D. R. Akbi, and N. Jawas: Proc. 2020 8th Int. Conf. Information and Communication Technology (ICoICT, 2020) 1–5. https://doi.org/10.1109/ICoICT49345.2020.9166329
8 S. Saidini, R. Haddad, N. Mezghani, and R. Bouallegue: Proc. 2018 Int. Conf. Smart Communications and Networking (SmartNets, 2018) 1–6. https://doi.org/10.1109/SMARTNETS.2018.8707391
9 S. Jeon, C. Lee, Y. Han, D. Seo, and I. Jung: Proc. 2017 IEEE Int. Conf. Consumer Electronics (ICCE, 2017) 108–109. https://doi.org/10.1109/ICCE.2017.7889246
10 W. Zhang, M. Tomizukam, and N. Byl: ASME J. Dyn. Syst. Meas. Control **138** (2016) 111004. https://doi.org/10.1115/1.4033949
11 K. Aoike, K. Nagamune, K. Takayama, R. Kuroda, and M. Kurosaka: Proc. 2016 IEEE Int. Conf. Systems, Man, and Cybernetics (SMC, 2016) 1257–1260. https://doi.org/10.1109/SMC.2016.7844414
12 C. Liu, J. Ying, F. Han, and M. Ruan: Proc. 2018 IEEE 3rd Int. Conf. Signal and Image Processing (ICSIP, 2018) 33–37. https://doi.org/10.1109/SIPROCESS.2018.8600483
13 S. Lee, S. M. Yoon, and H. Cho: Proc. 2017 IEEE Int. Conf. Big Data and Smart Computing (BigComp, 2017) 131–134. https://doi.org/10.1109/BIGCOMP.2017.7881728
14 A. S. A. Sukor, A. Zakaria, and N. A. Rahim: Proc. 2018 IEEE 14th Int. Colloquium on Signal Processing & Its Applications (CSPA, 2018) 233–238. https://doi.org/10.1109/CSPA.2018.8368718
15 K. J. Hunt, D. Hurwit, K. Robell, C. Gatewood, I. B. Boster, and G. Matheson: Am. J. Sports Med. **45** (2017) 426. https://doi.org/10.1177/0363546516666815
16 L. S. Wilson Jr, M. S. Mizel, and J. D. Michelson: Foot Ankle Int. **8** (2001) 649. https://doi.org/10.1177/107110070102200806
17 H. Schmal, A. H. Larsen, L. Froberg, J. L. Erichsen, C. F. Madsen, and L. Pedersen: Trials **20** (2019) 324. https://doi.org/10.1186/s13063-019-3447-8
18 S. Lamer, J. Hebert-Davies, V. Dube, S. Leduc, E. Sandman, J. Menard, and M. Nault: Orthop. J. Sports Med. **7** (2019) 2325967119864018. https://doi.org/10.1177/2325967119864018
19 W. Pirker and R. Katzenschlager: Wien. Klin. Wochenschr. **129** (2016). https://doi.org/10.1007/s00508-016-1096-4
20 C. Camathias, E. Ammann, R. L. Meier, E. Rutz, P. Vavken, and K. Studer: Knee Surgery, Sports Traumatology, Arthroscopy **28** (2020) 2053. https://doi.org/10.1007/s00167-020-05911-y
21 K. Liu, C. Hsieh, S. J. Hsu, and C. Chan: IEEE Sens J. **18** (2018) 9882. https://doi.org/10.1109/JSEN.2018.2872835
22 G. Wang, Q. Li, L. Wang, W. Wang, M. Wu, and T. Liu: Sensors **18** (2018) 1965. https://doi.org/10.3390%2Fs18061965

23 A. Dehghani, O. Sarbishei, T. Glatard, and E. Shihab: Sensors **19** (2019) 5026. https://doi.org/10.3390/s19225026
24 O. Dehzangi, and V. Sahu: 2018 24th Int. Conf. Pattern Recognition (ICPR, 2018) 1402–1407. https://doi.org/10.1109/ICPR.2018.8546311
25 R. A. Bhuiyan, Md. Amiruzzaman, N. Ahmed, and MD. R. Islam: 2020 3rd IEEE Int. Conf. Knowledge Innovation and Invention (ICIKII, 2020) 344–347. https://doi.org/10.1109/ICKII50300.2020.9318786
26 I. T. Jolliffe and J. Cadima: Philos. Trans. R. Soc. A **374** (2016) 20150202. https://doi.org/10.1098/rsta.2015.0202
27 B. Scholkopf, A. Smola, and K. Muller: Proc. Int. Conf. Artificial Neural Networks 1327 (ICANN'97, 1997) 583–588. https://doi.org/10.1007/BFb0020217
28 H. Zou, T. Haste, and R. Tibshirani: J. Comput. Graphical. Stat. **15** (2012) 265. https://doi.org/10.1198/106186006X113430
29 N. Golestani and M. Moghaddam: Proc. 2020 14th European Conf. Antennas and Propagation (EuCAP, 2020) 1–3. https://doi.org/10.23919/EuCAP48036.2020.9135215
30 T. Sawanglok, T. Thampairoj, and P. Songmuang: Proc. 2018 Int. Joint Symp. Artificial Intelligence and Natural Language Processing (SAI-NLP, 2018) 1–6. https://doi.org/10.1109/iSAI-NLP.2018.8692801
31 N. Hnoohom, A. Jitpattanakul, P. Inuluergsri, P. Wongbudsri, and W. Ployput: Proc. 2018 Int. ECTI Northern Section Conf. Electrical, Electronics, Computer and Telecommunications Engineering (ECTI-NCON, 2018) 111–115. https://doi.org/10.1109/ECTI-NCON.2018.8378292
32 T. Hayama and R. Arakwa: Proc. 2020 9th Int. Congr. Advanced Applied Informatics (IIAI-AAI, 2020) 400–403. https://doi.org/10.1109/IIAI-AAI50415.2020.00087
33 C. Hou: Proc. 2020 5th Int. Conf. Computer and Communications Systems (ICCS, 2020) 225–235. https://doi.org/10.1109/ICCCS49078.2020.9118506
34 M. S. H. Bhuiyan, N. S. Patwary, P. K. Saha, and M. T. Hossain: Proc. 2020 2nd Int. Conf. Advanced Information and Communication Technology (ICAICT, 2020) 286–290. https://doi.org/10.1109/ICAICT51780.2020.9333470
35 A. Wang, H. Chen, C. Zheng, L. Zhao, J. Liu, and L. Wang: Proc. 2020 Int. Conf. Networking and Network Applications (NaNA, 2020) 310–315. https://doi.org/10.1109/NaNA51271.2020.00060
36 R. Mutegeki and D. S. Han: Proc. 2020 Int. Conf. Artificial Intelligence in Information and Communication (ICAIIC, 2020) 362–366. https://doi.org/10.1109/ICAIIC48513.2020.9065078