**Methodology**

In this research, popular machine learning algorithms for Fall Detection were reviewed. The data were collected using the methods described in the article [1]. The methods used on that article are described here to make it easier to understand what we are doing. .

**Data Collection**

The method used the Gait Analysis framework that presented a comprehensive solution for addressing the fall issue. The authors selected 12 body key-points to represent human upper and lower limbs gait information. The gait key-points were detected from videos using MediaPipe [10]. The authors used Google’s innovative open source project MediaPipe [11] to present an assessment that was faster, simpler, more cost-effective, portable and easy to implement. This framework enabled developers to build multi-modal (video, audio and time-series data) cross-platform applied machine learning pipelines. A total of 12 landmarks: the left and right landmarks of the shoulders, elbow, wrist, hip, knee and ankle were tracked representing the movement of the upper and lower limbs. Extracting image sequences from the streaming video received via edge computing were passed to MediaPipe pose estimation for extracting those 12 landmarks movement. The landmark points obtained from the image sequences were time-series gait information, which were then passed for post-processing and normalization.

**Post-processing and normalization**

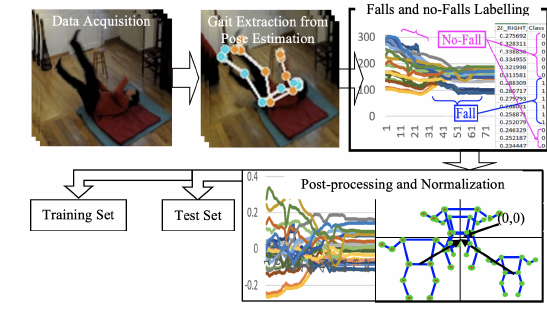
The collected 12 land-marks from the MediaPipe contain x, y and z coordinates. The authors considered only x- and y-coordinates in each image and ignored image frames with zero values. The landmark points of each image frame thought of as a closed shape. The person moving in the image sequence then showed the movement of the shape. To make the shape invariant of translation effect, the shape of each frame was translated to the center of the image frame. The shape consisted of landmark coordinates *X*i and *Y*i where i = 1, 2, 3. . . , n. Equations 1 and 2 were used to obtain the average values of *X*m and *Y*m.

|  |
| --- |
| n  *X*m = Σ *X*i (1)  i = 0 |
| N  *Y*m = Σ *Y*i (2)  i = 0 |

Equations 3 and 4 were applied to obtain the new coordinates c*X*i and c*Y*i of the translated shape which was at the center of the image frame.

|  |
| --- |
| *cX*i = *X*m - *X*i (3) |
| *cY*i = *Y*m - *Y*i (4) |
|  |

The width and height of the image frame were different for multiple sources of streaming videos. Therefore, the new coordinate values were divided by the corresponding width and height to normalize between 0 and 1. The landmarks with the image frame sequence represented a time-series of gait information. The time-series gait data were then stored in a csv file. The csv file contained the timestamp, *cX* and *cY* coordinates of 12 landmarks, a total of 25 columns. The csv file was then used for labeling fall and no-fall.



**Fall and no fall labeling**

The collected dataset included various activities and falls. The authors also considered the dataset falling or non-falling person. Falling was assigned to class 1 and non-falling to class 0. A part of the FDD dataset had a fall start frame number and a fall end frame number. The algorithm was designed to mark frames from start to finish as 1 and other frames as 0 in the csv file. Therefore, the last column (26th) of the csv file represented the classes.

Now after following the methods stated above we have followed the process below reviewing the performance of various machine learning models on the dataset.

**Dataset Splitting**

The dataset was divided into two parts: training and testing, we used 80% for training and 20% for testing.

**Feature Selection**

There were 12 landmarks as a feature and we used them all.

**Algorithms Used**

We analyzed the fall detection dataset gathered from paper [1] using six different machine learning algorithms: XGBoost, Random Forest, KNN, Logistic Regression, Perceptron, and Support Vector Machines (SVM). The dataset contained two classes: 0 for no fall and 1 for fall, and included various features related to body movement and position. The goal of the analysis was to develop a model that could accurately classify falls in real-time.

XGBoost is a popular gradient boosting algorithm that has been shown to be effective for a variety of machine learning tasks. The algorithm works by iteratively adding decision trees to the model, with each subsequent tree correcting the errors of the previous trees. In the fall detection dataset, XGBoost was trained on the features and labels in the training set, and the hyperparameters were optimized using cross-validation. The performance of the model was evaluated using various metrics such as accuracy, precision, recall, and F1-score.

Random Forest is an ensemble method that combines multiple decision trees to improve the accuracy and robustness of the model. The algorithm works by creating a set of decision trees, each trained on a random subset of the features and samples in the dataset. In the fall detection dataset, Random Forest was trained on the features and labels in the training set, and the hyperparameters were optimized using cross-validation. The performance of the model was evaluated using various metrics such as accuracy, precision, recall, and F1-score.

KNN is a non-parametric algorithm that works by finding the k-nearest neighbors of a given data point and classifying it based on the majority class of its neighbors. In the fall detection dataset, KNN was trained on the features and labels in the training set, and the value of k was optimized using cross-validation. The performance of the model was evaluated using various metrics such as accuracy, precision, recall, and F1-score.

Logistic Regression is a linear model that works by fitting a logistic function to the input features, which maps the input to a probability between 0 and 1. In the fall detection dataset, Logistic Regression was trained on the features and labels in the training set, and the hyperparameters were optimized using cross-validation. The performance of the model was evaluated using various metrics such as accuracy, precision, recall, and F1-score.

Perceptron is a simple neural network that works by using a single layer of perceptron units to separate the input into two classes. In the fall detection dataset, Perceptron was trained on the features and labels in the training set, and the hyperparameters were optimized using cross-validation. The performance of the model was evaluated using various metrics such as accuracy, precision, recall, and F1-score.

Support Vector Machines (SVM) is a supervised learning algorithm that finds a hyperplane in a high-dimensional feature space to separate the data into different classes. In the fall detection dataset, SVM was trained on the features and labels in the training set, and the hyperparameters were optimized using cross-validation. The performance of the model was evaluated using various metrics such as accuracy, precision, recall, and F1-score.

References:

[1] BIBLOTXT: @inproceedings{anwary2022deep,

title={Deep Learning enabled Fall Detection exploiting Gait Analysis},

author={Anwary, Arif Reza and Rahman, Md Arafatur and Muzahid, Abu Jafar Md and Ashraf, Akanda Wahid Ul and Patwary, Mohammad and Hussain, Amir},

booktitle={2022 44th Annual International Conference of the IEEE Engineering in Medicine \& Biology Society (EMBC)},

pages={4683--4686},

year={2022},

organization={IEEE}

}

Actual cite: Anwary, Arif Reza, et al. "Deep Learning enabled Fall Detection exploiting Gait Analysis." 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE, 2022.