Gait Classification of Parkinson's Disease Patients Using Efficient Representations From Autoencoders

Autor:
Juan Sebastian Guerrero Cristancho

Degree work, requirement to opt for the title of:
Electronics Engineer

Advisors:
Prof. Dr.-Ing. Juan Rafael Orozco Arroyave
Paula Andrea Pérez Toro MSc.

Research line:
Pattern Analysis & Signal Processing
GITA Research Group

Universidad de Antioquia
Faculty of Engineering, Department of Electronics Engineering and Telecommunications.
Medellín, Colombia

2022.

Cita	(Guerrero Cristancho, 2022)
Referencia	Guerrero Cristancho, J. S. (2022). Gait *Classification of Parkinson's Disease Patients Using Efficient Representations from Autoencoders,* Trabajo de grado. Universidad de Antioquia, Medellín, Colombia.
Estilo APA 7 (2020)	

Grupo de Investigación en Telecomunicaciones Aplicadas (GITA).

Content Table

Abstract

Parkinson's Disease (PD) is one of the most common neurodegenerative diseases. Patients manifest a progressive degeneration of dopamine, which plays a key role in abilities such as the locomotion, cognitive capabilities, sleep regulation and mood. One of the symptoms of the disease is the progressive gait impairment, resting tremors, slowness of movement, shuffling steps, among others. There is interest among the scientific community to develop automatic classification systems to support the diagnosis. The goal is to properly discriminate the disease and to predict the neurological state of the patients. This work focuses on the use of Convolutional Auto-Encoders to obtain efficient representations from multi-channel gait signals from Smartphones and sensors to classify PD patients vs. Healthy subjects. The channels represent the acceleration in the 3-dimensional plane (X, Y, Z). The proposed experiments consist of three models using 64, 128, and 256-dimensional bottlenecks to compress the information of gait signals. The accuracy and unweighted average recall are used to evaluate the classification performance over the PC-GITA database, from which 38 controls and 38 subjects were used for training the neural networks, and 30 patients and healthy subjects were used as test dataset. The subjects were asked to perform the 4x10 gait task, which consists of four repetitions of walking for 10 meters, stop and perform a 180° turn. A Stratified 5-Fold-Cross-Validation strategy is used to evaluate the performance of a Support Vector Machine over the testing dataset. The results indicate that the 64-dimensional bottlenecks provide enough information to properly differentiate between patients and controls. The results report accuracy of up to 85%, and Unweighted average recall values of 93%. Additionally, the area under the ROC curve is reported for each fold. There is no variation in the results when considering gait signals with non -randomized and randomized channels. It is concluded that the methodology is suitable to classify patients vs. healthy subjects, despite the different origins from the signals and the challenges that different sampling frequencies impose for such a methodology.

Introduction

Parkinson's Disease (PD) is a degenerative condition which affects the correct functioning of the brain and the motor system [1]. Patients manifest symptoms such as bradykinesia, tremors in the limbs and impaired gait. Gait disorders affect the patients' rhythm control, symmetry, coordination, and postural stability [1]. The aforementioned affect dramatically the quality of life of the patients [2].

Cognitive deficits in PD consist of a series of events that lead to the progressive loss of the neurons which produce dopamine. These are located within the substancia nigra, in the basal ganglia. Dopamine is a critical neurotransmitter that plays key roles in the locomotion, cognition capability, sleep regulation, humor, the ability of learning, among others. The direct consequences in gait include festinating gait, shuffling steps, slowness of movement, freezing of gait, among others. The indirect consequences include depression and sleep disorders. The treatment for PD involves the active consumption of levodopa, a compound designated to compensate for the loss of dopamine in the body. Gait therapy has achieved small yet important contributions in gait rehabilitation for PD patients. Parkinsonian gait is treated with cognitive therapy, musculoskeletal and cardiorespiratory therapy, and the promotion of physical activities oriented to the state of each patient [3]. There is evidence that during the early to mild stages of PD, patients are still able to learn new motor skills [4]. Other studies report that combined task programs have proved to be efficient in walking therapy for PD patients with mild to moderate affection [5-6].

The analysis of parkinsonian gait is conducted by assessing the kinematic, functional, and postural capabilities of the patients. According to the Movement Disorder Society-Unified Parkinson's Disease Rating Scale Part III (MDS-UPDRS-III) [7], kinematic aspects include the agility of the legs and stepping rate. Functional capabilities comprise the performance and quality of the walking process. The postural capabilities are related to the patients' stability when standing up. In the last decade, there has been a lot of enthusiasm towards data-driven machine learning applications. These are among the most interesting paradigms, products of the union between medicine and technology [8-9-10]. Artificial intelligence (AI) is the most important trend in technology in the last decade. Medical research using AI techniques focuses primarily on classification tasks, such as PD patients vs.

Healthy Control (HC) subjects [11-12-13]. The data is processed to reduce redundant and irrelevant information. Then, the feature extraction process is performed. The last step involves training and validation of a classifier to discriminate samples.

The objective of this work is to classify between PD patients and HCs by implementing a Neural Network (NN) architecture composed of AutoEncoders (AE) to extract features from gait signals, and a Support Vector Machine to classify PD patients vs. HCs. The main contribution of this work is to use two data sources simultaneously (accelerometer data from smartphones, and data from shoe-attached sensors), to obtain compressed representations with an AE. Such representations will be used to classify PD patients vs. HCs. This is one of the few approaches using this technique and could be relevant to deploy small and precise systems for the automatic detection of PD.

1. Objectives

2.1. General Objective

- Design, implement and test a methodology based on gait signals and AEs to classify between PD patients vs. HC subjects.

2.2. Specific Objectives

- To extract and analyze features from AEs to model gait signals captured from external sensors and smartphone accelerometers.
- To design and implement a system to classify PD patients vs. HC subjects using gait signals.
- To use automatic selection methods to evaluate and select the most suitable features to improve the classification process.
- To validate the proposed algorithms, the performance will be evaluated according to different metrics, such as Accuracy (ACC), and the Area Under the ROC curve (AUC).

3. Theoretical Background

This section covers the necessary methods to fulfill the aforementioned objectives.

3.1. Gait Signals Processing

3.1.1. e-Gait shoe attached sensors

The eGait sensors[1] are composed of a 3-dimensional accelerometer, and a 3-dimensional gyroscope. Each sensor is attached to the upper right heel of each shoe. Each accelerometer and gyroscope feature a range of ±6g and (5)00°/s respectively. Signals are recorded in a tablet, via Bluetooth. A proprietary Android application (app) allows the user to select specific tasks and ingress relevant information of the patient. These sensors have a sampling frequency of 100Hz, with a resolution of 12 bits. Figure 1 shows an eGait sensor attached to a shoe.

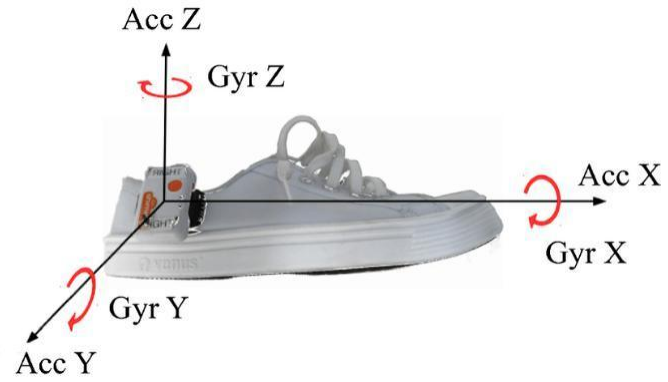

Figure 1. eGait sensor. Source [12].

3.2.2. Apkinson Android App

Apkinson is an Android application to evaluate movement and speech impairments of PD patients. It includes exercises designed to evaluate individual muscle groups, such as hands, arms, and legs. These exercises are randomly selected from various groups, in such a fashion that ensures one exercise per group in the session. Upon finishing a session, the patient's performance is evaluated and the data is stored for further analysis from previous sessions. The source code of the application is available in a Github[2] repository. This application serves from the smartphone's built-in accelerometer, and its sampling frequency ranges from 2-200Hz, depending on resource availability. The biggest drawback from this method is the way the X-axis, Y-axis, and Z-axis are assigned in the accelerometer, with significant variations from one manufacturer to another. For the purpose of this work, all 3 channels are randomized for each recording [10].

[1] Embedded Gait analysis using Intelligent Technology, http://www.egait.de/
[2] https://github.com/SAGI-FAU/SMA2

3.1.2. Resampling [19]

It is not clear that the smartphones can contain different sampling frequencies, due to the power saving and app managing. Each smartphone does not have a fixed sampling frequency. In order to solve this issue, the signals from both sources are resampled using a quadratic interpolation. It is applied to both sources, matching the sampling frequencies to 50 Hz.

Let (x_0, y_0), (x_1, y_1), and (x_2, y_2) be three points, which are considered in order to obtain a second-degree polynomial $P^2(x) = a + bx + cx^2$. For such a task, all the coefficients must be obtained. In this case, there is exactly one polynomial of degree ≤ 2 that interpolates the $n + 1$ data points. The condition $f(x_i) = y_i \, for \, 0 \leq i \leq n$ can be rewritten as a linear system, in terms of the coefficients of $P^2(x)$ as follows:

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}$$

The linear system $V\boldsymbol{b} = y$, where $\boldsymbol{b} = (b_0, b_1, \dots, b_n)$ is the vector of coefficients, $y = (y_0, y_1, \dots, y_n)$ is the target vector, and V is the Vandermonde matrix. This system could be solved using various methods. For simplicity, the Lagrange Interpolation method is used in this work. Let the functions $L_0(x)$, $L_1(x)$, $L_2(x)$ be defined as follows:

$$L_1(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \qquad (1)$$

$$L_2(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \qquad (2)$$

$$L_3(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \qquad (3)$$

The interpolation equation $P_2(x)$ is defined in Equation 4.

$$P_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) \quad (4)$$

The process is performed by replacing the data points from the gait signals in Equation 4 to obtain sections that are used to sample a signal with a frequency of 50 Hz. Figures 3-4 illustrate the result.

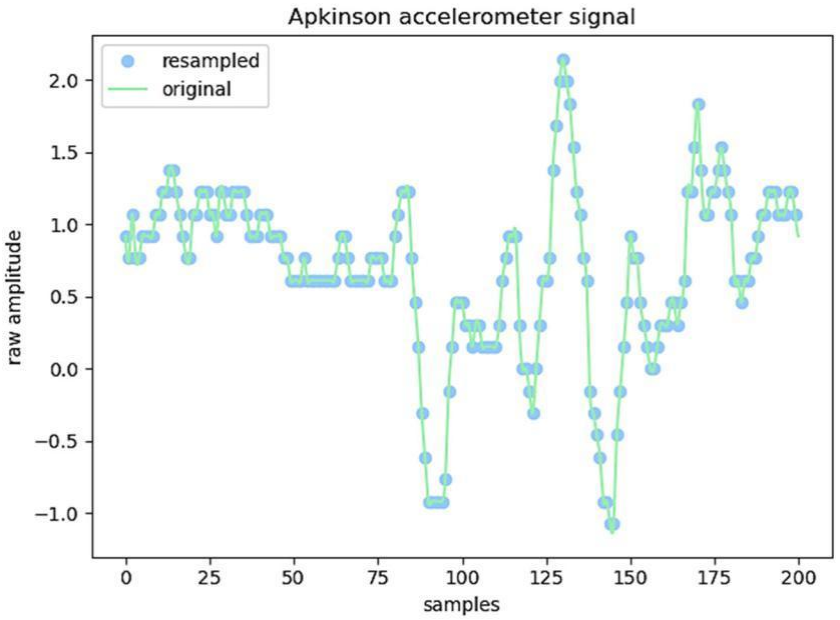

Figure 3. Apkinson signal resampled to 50Hz. Source: Author.

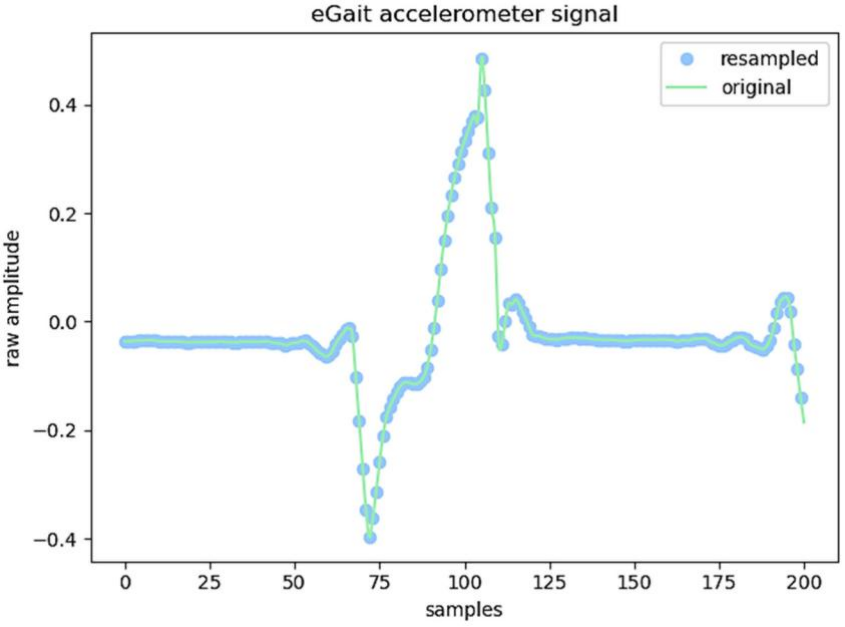

Figure 4. eGait signal resampled to 50Hz. Source: Aut

3.2. Classification

3.2.1. Support Vector Machines [20]

A Support Vector Machine (SVM) is a supervised learning model which differentiates classes by finding the optimal hyperplane that better separates the classes. The idea is to transform the data points into linearly separable data. This algorithm uses a kernel function to perform such transformations. Let be the feature vector , , … , and the labels for each , , , … , . The closest points to the hyperplane are the support vectors, and belong to different classes. The distance between such vectors is the margin. Figure 4 illustrates the idea in a hard margin SVM, which uses the closest points from each class to compute the optimal hyperplane. Hard margin SVMs do not allow points from different classes to be placed inside the margin, on the other hand, soft margin SVMs allow such scenarios and are less sensitive to outliers.

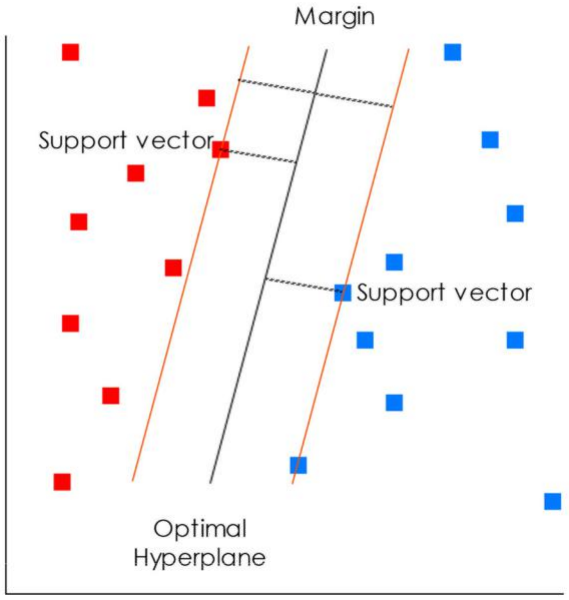

Figure 4. Hard margin SVM. Source: Author.

The optimal hyperplane is the one with the maximum margin. The optimization process is as follows: Let $\phi(X_n)$ be the kernel function, which transforms the feature space of X_n into a space where the features are linearly separable. W is the weight vector, is b is the bias term, ζ and is a slack variable which defines a margin of tolerance for errors. The Radial-Basis-Function (RBF) Kernel and the linear kernel will be considered for this work. The decision function is described in Equation 5.

$$y_n[W^T\phi(X_n) + b] \geq 1 - \zeta_n \; \forall \in \{1, \ldots, N\} \quad (5)$$

Where is the complexity parameter that controls the balance between the width of the margin and . defines the radius of the RBF. The RBF kernel is based on a normal distribution with standard deviation (std) and is defined as follows in Equations 6 and 7.

$$\gamma = \frac{1}{2\sigma^2} \quad (6)$$

$$\phi_{RBF}(X_n) = e^{-\gamma||X_n - X_m||^2} \quad (7)$$

The linear kernel is defined as the inner product between X_n and X_m added to a constant k.

$$\phi_L(X_n) = X_n^T X_m + k \quad (8)$$

This work covers a bi-class scenario; therefore, the objective function to minimize is:

$$minimize \; \frac{1}{2}||W||^2 + C\sum_{i=1}^{n}\zeta_i \quad (9)$$

3.2.2. Area under the ROC Curve [21]

The Receiver Operating Characteristic curve (ROC) is a tool designed to evaluate the predictions of a binary classifier. Such a tool was originally named due to its creation and use in military radar receivers. The graph is calculated by computing the True Positive Rate (TPR), and the False Positive Rate (FPR) for a range of decision thresholds. Equations 10 and 11 define these concepts.

$$TPR = \frac{False\ Negatives}{True\ Positives} \qquad (10)$$

$$FPR = \frac{False\ Positives}{True\ Falses} \qquad (11)$$

In a ROC curve, the axis is defined as FPR, and the axis as the TPR. The Area under the curve (AUC) is the area covered by the resulting curve of the ROC curve. A classifier performs worse the closer the AUC value is to 0.5, which tells that it is no better than random guessing. Figure 5 Illustrates the idea.

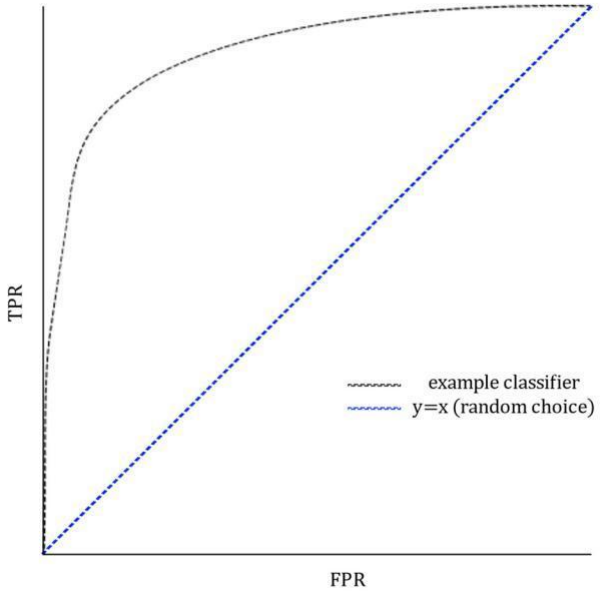

Figure 5. ROC Curve example. Source: Author.

3.2.3. Confusion Matrix [21]

This is a very popular technique to evaluate the performance of a classifier. This work considers a bi-class scenario; therefore, four coefficients are computed: True Positives (TP). False Negatives (FN), False Positives (FP), True Negatives (TN). These coefficients are ordered in a fashion illustrated by Table 1.

Real		Predicted	
		Positives	Negatives
	Positives	TP	FN
	Negatives	FP	TN

Table 1. Confusion Matrix. Source: Author.

3.2.4. K-Fold Cross Validation [21]

The performance of a model over limited data is measured using Cross-Validation (CV) techniques. In a K-Fold CV, the parameter is the number of folds in which the data will be split. In this process, a total of folds are used for training, and testing. The process is iterative and in each fold the data is divided as follows: k-1 folds are used for training, 1-fold is used for testing. For each fold, a Grid Search hyperparameter tuning is performed. The best hyperparameters are selected according to the accuracy in the test datasets. A 5-Fold CV is performed, and the process is illustrated in Figure 6.

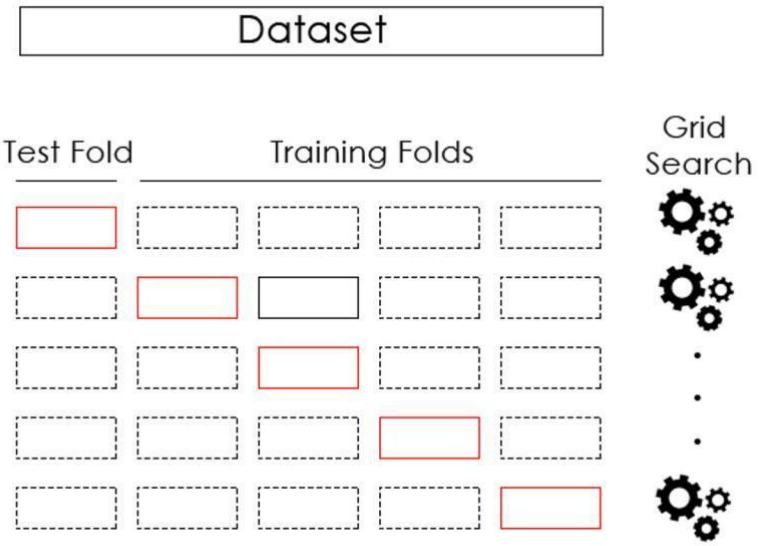

Figure 6. 5-Fold Cross-Validation. Source: Author.

Parameter (SVM)	Values
C	10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10
γ	10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1
Kernel	Linear, RBF

Table 2. Hyperparameter grid.

3.3. Neural Networks

3.3.1. Feed-forward Neural Networks [23]

In this architecture, the information flows in only one direction: from the input of the NN, to the output. There are X_D inputs, k hidden layers H_K, and n outputs Z_N. There are sets of weights $W_{k-1,\,k}$ that leverage the connections between the layers. Those are composed of neurons, which are represented by activation functions $\sigma_k(W_{k-1}, H_k)$ that respond to the product of the weights added to the sum of biases from previous layers. Activation functions are selected according to the layer type, task among others. For instance, for image recognition, functions with no response to negative values are often selected e.g Rectifier Linear Units (ReLU).

Let $W^T_{0,1} * H_0$ be the product between the input layer of the network and the $\mathbf{0}$,$\mathbf{1}$ and the weights associated between it and H_1. The first activation function output will be $H_1 = \sigma_1(W^T_{0,1} * X + b)$ where b is a bias term which can be randomly initialized. The output in subsequent layers will be described by Equation 12. Figure 7 illustrates the main structure of this architecture.

$$H_k = \sigma_k(W_{k-1,k'}\, H_{k.1}) \quad (12)$$

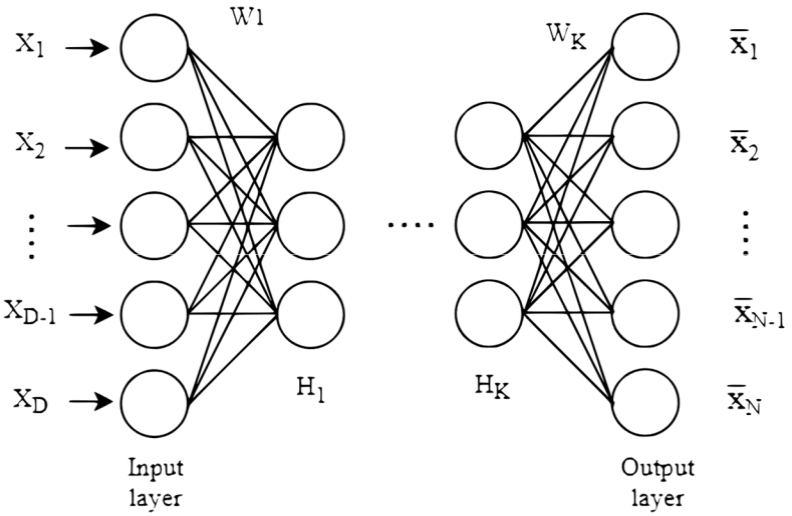

Figure 7. Architecture of a feed-forward NN. Source: Author.

3.2.2. Autoencoders [24]

This architecture aims to learn efficient data representations to reconstruct the X inputs of the NN. In this architecture, $\underline{X}_D$ are outputs, the H_k hidden layers compress the structure of the data and produce a representation of its structure. The optimization process focuses on minimizing the reconstruction error $ς(X, \underline{X})$, which is the distance between the original input and its representation inside the net. The reconstruction error is chosen conveniently, according to a task. For instance, the Mean Squared Error (MSE) is typically used in AEs, as shown in Equation 6. The motivation behind AEs is that non-linear relationships and manifolds could be learned, contrary to traditional techniques such as Principal Component Analysis. The weight of the layers is adjusted according to ς. Figure 8 showcases this architecture.

$$ς(X, \underline{X}) = \frac{1}{D}\sum_{i=1}^{D} ς(X_i, \underline{X}_i) \quad (13)$$

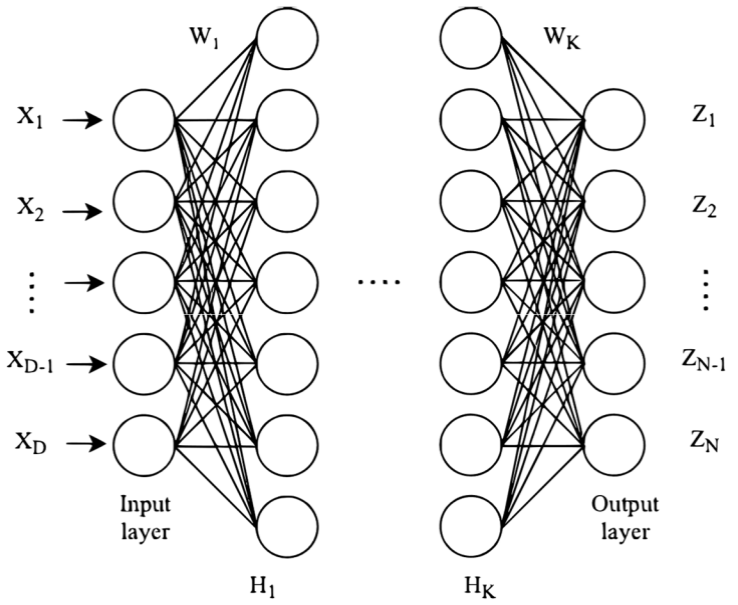

Figure 8. AE Architecture. Source: Author.

3.2.2. Long Short-Term Memory [25]

Recurrent Neural Network (RNN) architectures allow models to operate considering temporal dynamics. Long Short-Term Memory (LSTM) cells introduce feedback connections and introduce an internal state to represent temporal information. LSTMs are composed of input, output, and forget gates, which regulate the flow of information through the cell. Let C_t be the cell state that depends on the instant t, σ is a sigmoid activation function, X_t the input in the instant t, h_{t-1} the output of a layer from the previous state, h_t the output at the current state, f_t the forget gate, i_t the information of the input gate, and o_t the output gate.

Figure 9 shows the internal architecture of a LSTM cell. The process begins with X_t flowing with the information from the previous layer h_{t-1}. f_t decides if the information of previous states will influence the next state. i_t will decide which information is going to be selected and a vector of candidates $C_t^{'}$ is computed by the $tanh$ function. o_t writes the next state C_t and will produce the connections to adjacent layers connected to the LSTM cell.

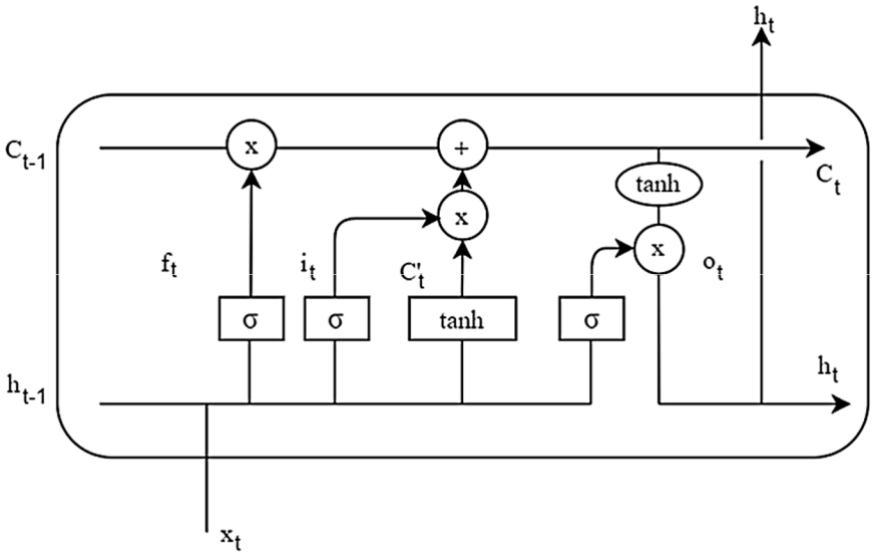

Figure 9. LSTM architecture. Source: Author.

The W_i weights depend of previous layers h_{i-1} and inputs X_i. Each layer has its own bias. Equations 13-15 describe the operation of these gates.

$$i_t = \sigma(W_i[X_t, h_{t-1}] + b_i) \quad (13)$$
$$f_t = \sigma(W_i[X_t, h_{t-1}] + b_f) \quad (14)$$
$$o_t = \sigma(W_i[X_t, h_{t-1}] + b_o) \quad (15)$$

In a similar fashion C_t is a function that depends on the previous layer and weights in a given timestamp. In order to update C_t, a candidate $\underline{C}_t$ is calculated in each timestamp, according to Equation 16. The state is updated in Equation 17.

$$\underline{C}_t = tanh(W_i[X_t, h_{t-1}] + b_c) \quad (16)$$
$$C_t = f_t * C_{t-1} + i_t * \underline{C}_t \quad (17)$$

Standard RNNs learn temporal sequences, however, it can be difficult for them to learn long-time temporal sequences. LSTMs solve this problem with the state gates, which control the flow of information over time. Gated Recurrent Units (GRUs) are simpler architectures than LSTMs. However, this work will include the use of LSTMs due to their high popularity in the scientific community [14-15-16].

3.2.3. Convolutional Neural Networks [36]

Convolutional Neural networks (CNN) are used in image processing as filters to learn feature maps. This work considers convolutional layers to extract features from the gait signals. The convolutional maps are computed by multiplying a sliding kernel matrix, with the inputs of the layer. Each connection learns a weight, associated from one layer to another. This type of NN operates usually along 2 dimensions, which are the width, and the height of an image. This work considers 1-D Convolutional layers, which are used to operate time series. Figure 10 illustrates the idea.

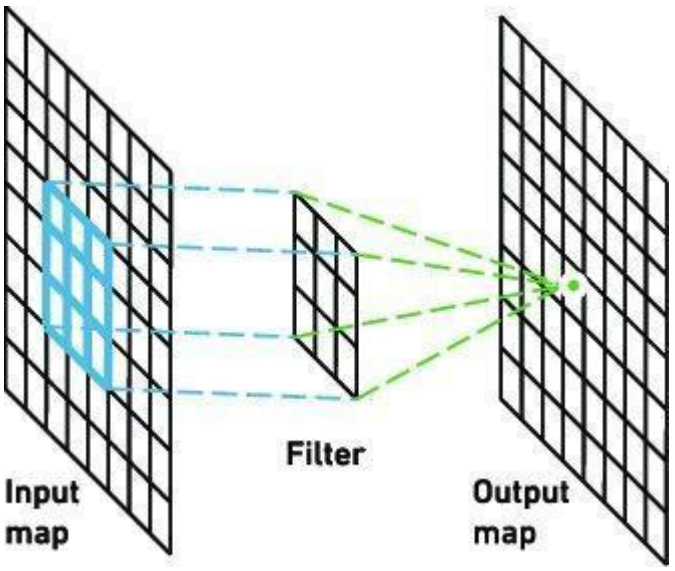

Figure 10. Convolutional layer. Source [33].

4. Data

The Dataset is composed of two parts: training and testing. The training data-set consists of 38 patients and 38 HCs. The testing data set has 30 patients and 30 controls. All Patients performed the 4x10 gait task. Additionally, all PDs and HCs had both Apkinson and eGait recordings, which were considered for the experiments. The inclusion criterion for the controls is to be absent of any movement disorder, impairment, or injury. Table 3 contains the demographical information for both groups. All subjects belong to a study conducted by GITA Lab, Universidad de Antioquia, Medellín, Colombia, and signed an informed consent, where they can opt out of the study anytime. The histogram of age distribution is illustrated in Figures 11-11b.

	PD patients	HC subjects
Gender [F/M]	14/16	18/12
Age [F/M]	58.2 ± 8.7 / 70.7 ± 7.1	60 ± 3.6 / 58.6 ± 4.1
Educational attainment[F/M]	10.5 ± 4.9 / 8.1 ± 5.2	-
Years since diagnosis [F/M]	10.5 ± 4.9 / 8.1 ± 5.3	-

Table 3. Demographic and clinical data from the testing dataset.

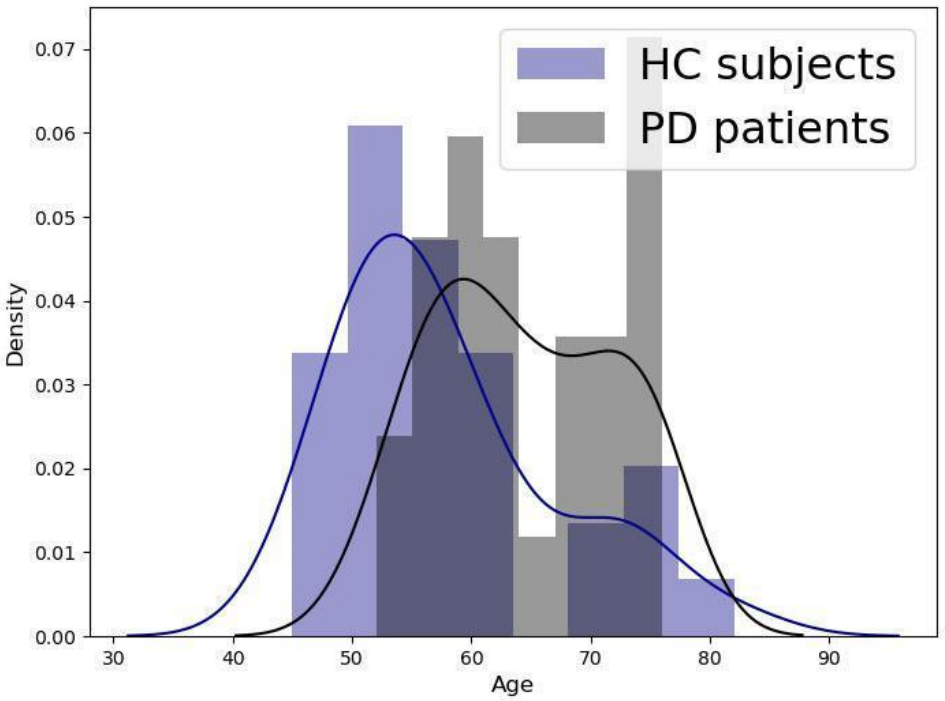

Figure 11. Age distribution of training dataset for CAENC. Source: Author.

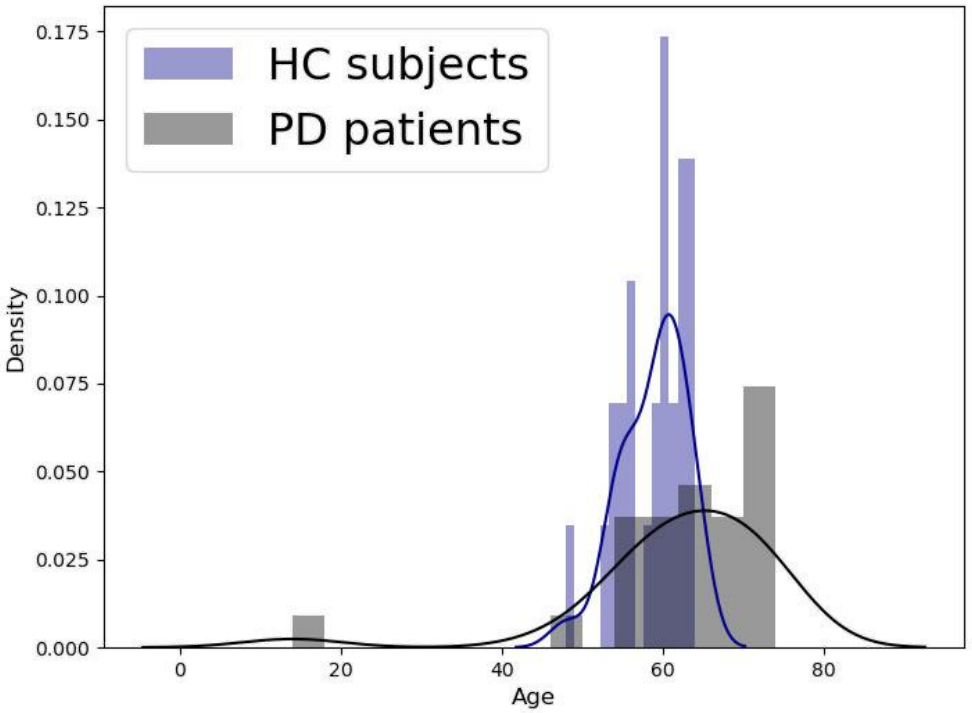

Figure 11-b. Age distribution of the test dataset. Source: Author.

5. Methodology

5.1. Model Training

The gait signals from the training dataset are used for this process. 70% of the data is used for training, 30% for validation. The training dataset purpose is to avoid biases that could result from combining information from the subjects of the testing dataset. Each recording has over 600.000 registers. Each patient recording is split in 3 channel windows, composed of 125 samples (2.5 seconds). These samples are used to train a convolutional AE (CAE).

The encoder is composed by a three-channel convolutional layer, which receives the three channel inputs and scales those channels by a factor of 2. A MaxPool1D layer with a kernel size and stride of 2 is used afterwards to select relevant features from the convolutional maps, followed by a Gaussian Error Linear Unit (GELU) activation function. The next stage is composed of three LSTM layers, which are aimed to learn features from the time variation of the signals. Next, a BatchNorm1D layer is added to normalize the outputs. The compression stage is composed of a sequential reduction of size composed of fully connected layers, which reduce 16128 neurons into 64, in 5 steps. Each step is a fully connected layer with dropout in order to reduce overfitting, as follows:

1. 16128 > 4096 Dropout: 20%
2. 4096 > 2048 Dropout: 20 %
3. 2048 > 1024 Dropout: 25 %
4. 1024 > 128 Dropout: 25 %
5. 128 > 64

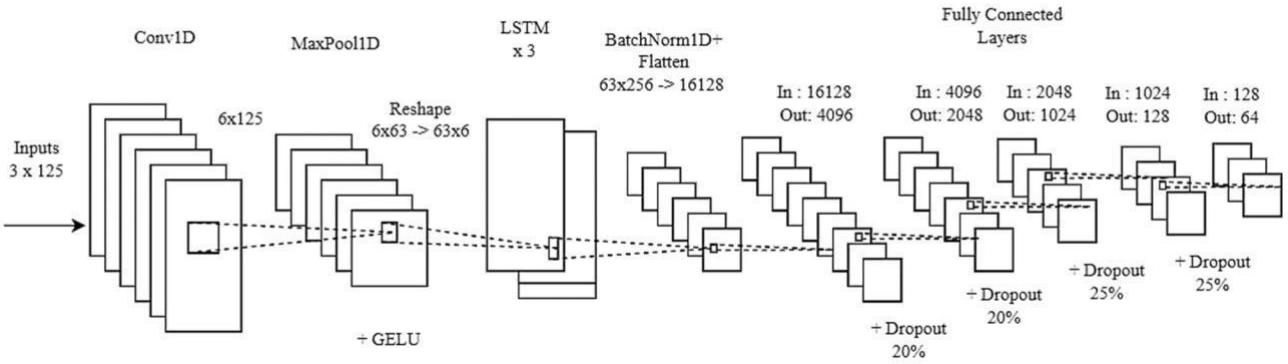

Figure 12. Encoder stage. Source: Author.

Figure 12 features the full connection of the encoder. The decoder follows a decompression stage, following 5 steps, which correspond to the inverse of those in the compression stage, with a variation of the dropouts between the fully connected layers (Fig. 13). The last fully connected layer is followed by a Sigmoid activation function. The outputs are fed into a three-layer LSTM with BatchNorm1D. An interpolation is computed in order to restore the same dimensions as the input and finally, a Convolution from 6 channels to 3 is performed, in order to compare the result with the original signal. Figure 13 illustrates the decoder stage.

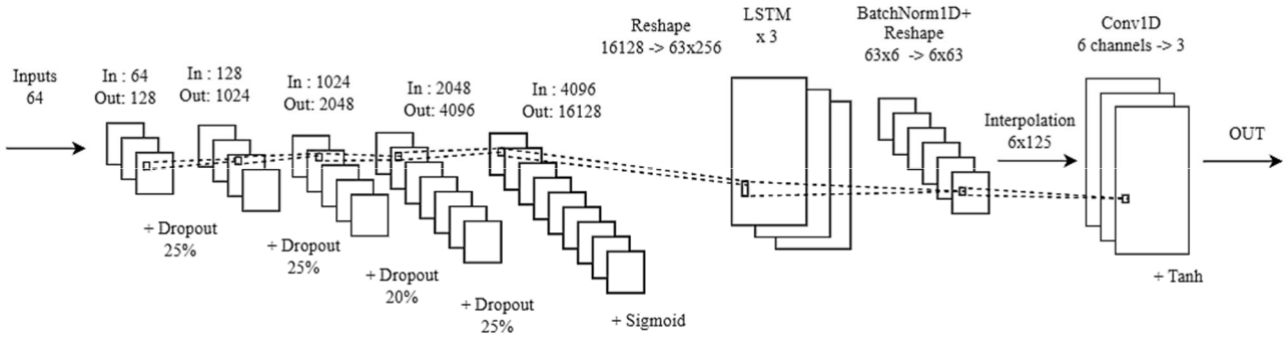

Figure 13. Decoder stage. Source: Author

The 64-neuron representation is the bottleneck of the network, which contains the compressed representation of the 3 channels. Two additional models are considered, using a 128 and 256-dimensional bottleneck respectively. This work is performed using Pytorch v1.11.0. The training is composed of the following parameters:

1. Batch size = 16
2. Epochs = 15
3. Training samples: 13739
4. Validation samples: 6767

The model is trained using the MSE loss function, and the Adam optimizer, with a learning rate of $1e^{-4}$, and weight decay (L2 regularization in Pytorch implementation)[3] of $1e^{-4}$. Figure 14 illustrates the loss summary for the best model. The loss values indicate that the errors in the output are small, in spite of the oversampling step in the decoder stage.

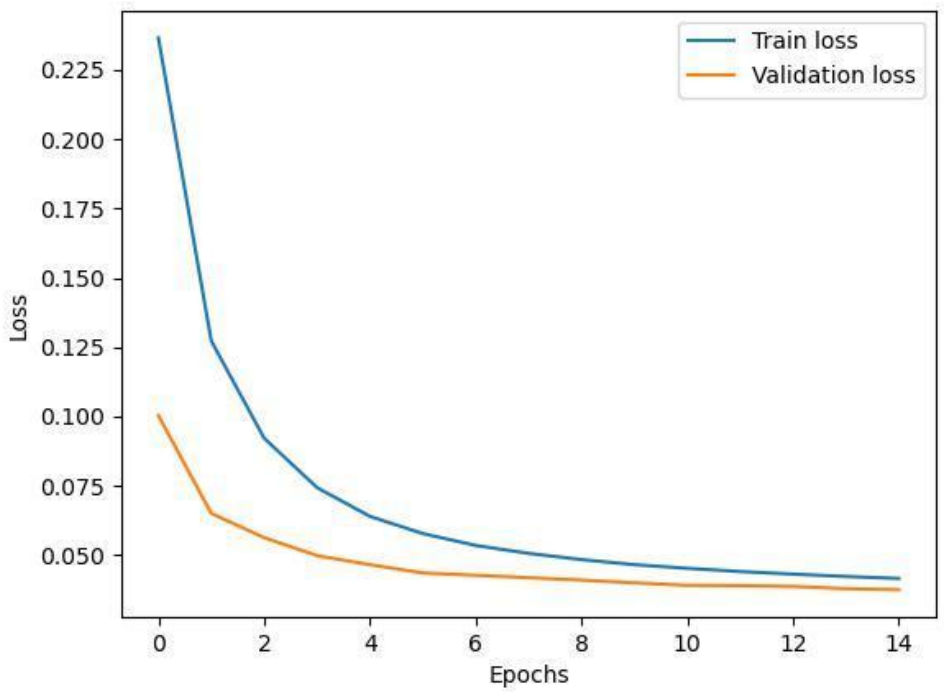

Figure 14. Training summary. Source: Author.

The model is trained considering the early stopping technique as a regularization method to prevent overfitting. Figures 15-17 show the comparison between the X, Y, and Z channels, which will now be considered as 1, 2, and 3, at the input of the network vs. the same channels at the output of the model. All plots correspond to a single random sample from the testing dataset.

[3] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html

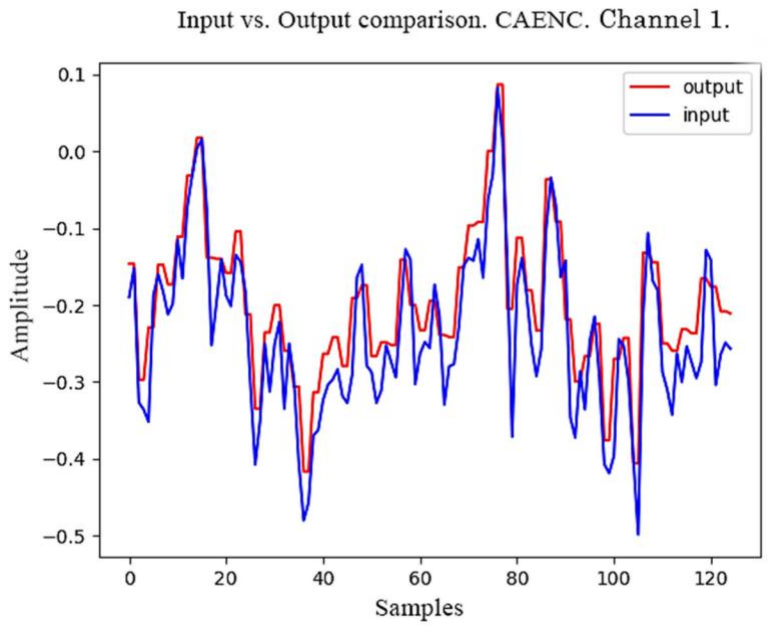

Figure 15. Random sample from channel 1. Source: Author.

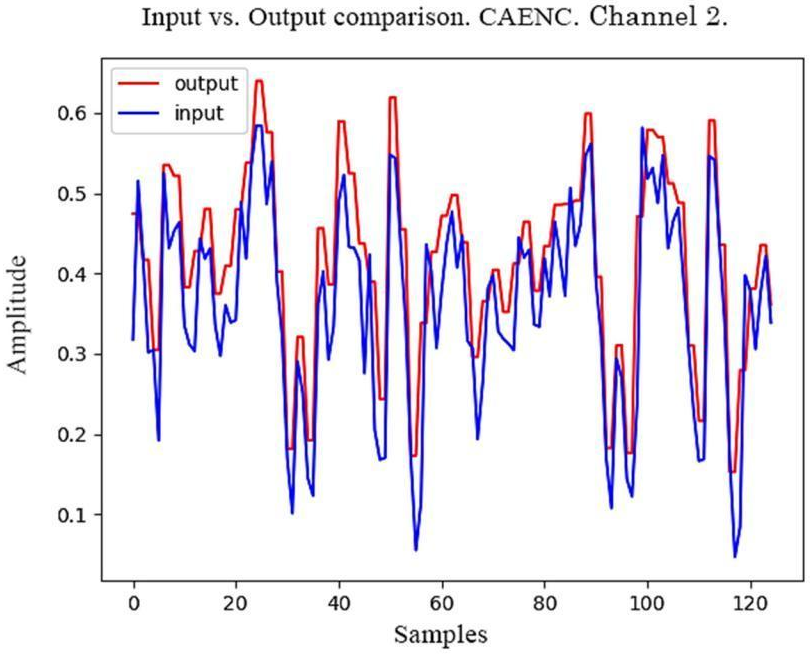

Figure 16. Random sample from channel 2. Source: Author.

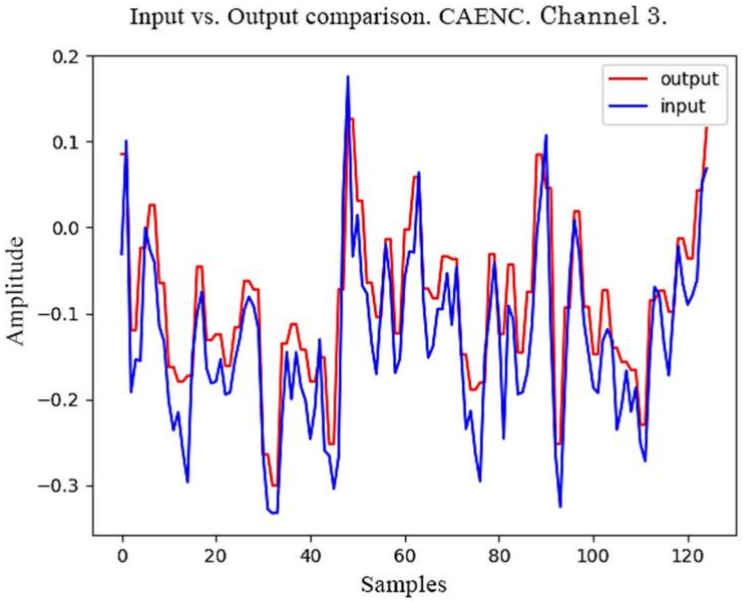

Figure 17. Random sample from channel 3. Source: Author.

The representation of the input follows the expected behavior, albeit with loss of information due to the interpolation. A Min-Max scaling process was computed per channel for the training process.

5.2. Feature Extraction

The model produces compressed representations from the 3 channels in the bottleneck section, which will be used as features. Three models are proposed for this work:

1. Model #1: 64-Dimensional bottleneck
2. Model #2: 128-Dimensional bottleneck
3. Model #3: 256-Dimensional bottleneck

Two statistical functionals are computed for each recording: the mean, and the std along the vertical axis of the feature vectors. The early fusion of these features is considered, forming a 128-dimensional vector per patient in #1, a 256-dimeinsional vector in #2, and a 512-dimensional vector for #3.

6. Experiments, results, and analysis

Three sets of experiments are conducted. First, randomized input channels are considered, and secondly, non-randomized channels. For each set, stratified 5-Fold-CV strategies are conducted, considering the scenarios listed in Table 5. The ROC curve plots for each experiment, ACC, Sensitivity (SEN), Specificity (SPEC), and Unweighted Average Recall (UAR) are reported in Tables 6 and 7.

The best hyperparameters are selected based on the ACC in the development set.

Four experiments are proposed. First, only signals from Apkinson are considered to classify PDs vs. HCs. The second experiment consists of the classification using the eGait system, left and right side are considered separately. The third experiment uses the early fusion of both sides for the classification task. The fourth experiment is only considered for the first model, since it is not possible to use different sources as it will introduce biases in the classifier when differentiating PDs vs HCs. Table 5 resumes the experiment list:

#	Side	Experiment		
1		PD Apkinson	VS.	
2	L	PD left foot eGait	VS.	HC left foot eGait
	R	PD right foot eGait	VS.	HC right foot eGait
3		PD eGait Early-Fusion Left-Right	VS.	HC eGait Early-Fusion Left-Right
4	**L**	**PD Apkinson**	**VS.**	**PD right foot eGait**
	R	**PD Apkinson**	**VS.**	**PD left foot eGait**

Table 5. Experiment list.

Experiments in sections **6.1-6.3** were conducted using randomized channels for the CAE. On the other hand, section **6.4** contains experiments using non-randomized channels.

6.1. Model #1 (64-dimensional bottleneck)

Experiment	ACC (%)	SEN	SPEC	UAR	Best Model
1	48.00 ± 9.67	0.15 ± 0,20	0.52 ± 0.23	0.33	C=0.001, gamma=0.001, kernel= RBF
2	80.30 ± 11.40	0.90 ± 0.13	0.93 ± 0.08	0.91	C=1, gamma=0.1, kernel= RBF
	82.20 ± 16.01	0.87 ± 0.19	0.93 ± 0.08	0.90	C=1, gamma=1, kernel=RBF
3	85.00 ± 6.20	0.93 ± 0.08	0.93 ± 0.08	0,93	C=1, gamma=1 kernel= RBF
4	79.60 ± 11.00	0.86 ± 0.17	0.86 ± 0.16	0.86	C=1, kernel=Linear
	83.10 ± 8.00	1.00 ± 0.00	1.00 ± 0.00	1.00	C=1, kernel=Linear

Table 6. Results for Model #1. 64-dimensional embeddings.

6.1.1. Experiment 1

The results from experiment 1 indicate that the proposed hyperparameter grid, classifier and features are not adequate to properly differentiate between classes. Another CV is performed using the mean, std, kurtosis and skewness as features. The results improved, resulting in the following metrics: **ACC** 71.3 ± (6.1), **SEN** 0.68 ± 0.06, **SPEC** 0.66 ± 0.06, and **UAR** 0.67. One of the explanations of such a result is that the AE models were not trained with signals from both sources, creating the necessity for more information. However, the consideration of more statistical functionals resulted in better results. Figure 18 illustrates the results for the experiment listed in Table 6.

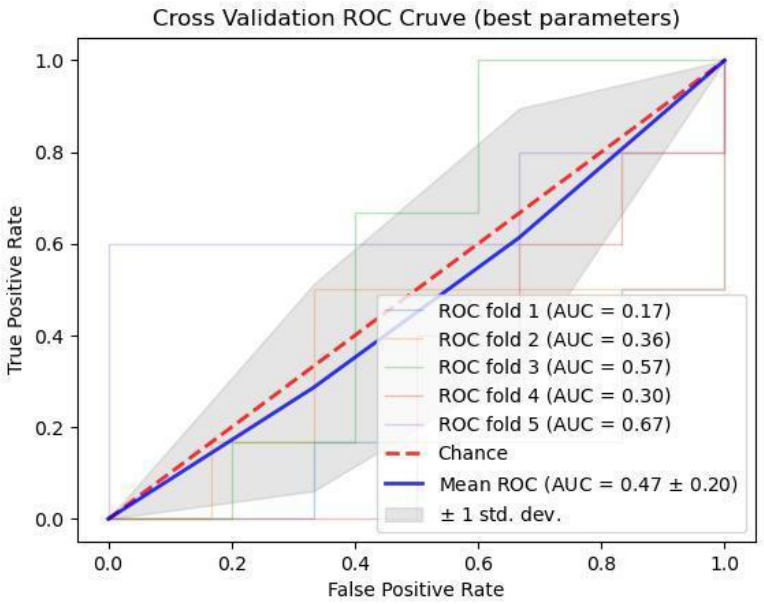

Figure 18. ROC report for Model #1 - experiment 1. Source: Author.

6.1.2. Experiment 2

This experiment showcases the comparison between the left and right sides. Although the ACC score is better than the results of Experiment 1, the stability of the models is inferior, according to the ROC Curve report in Figures 19-20, and Table 6. The Same applies to the SEN, and SPEC values. Additionally, the complexity parameters for the best models are equal, with the difference being in the gamma parameter, which means that the decision boundary is more curved in the left side.

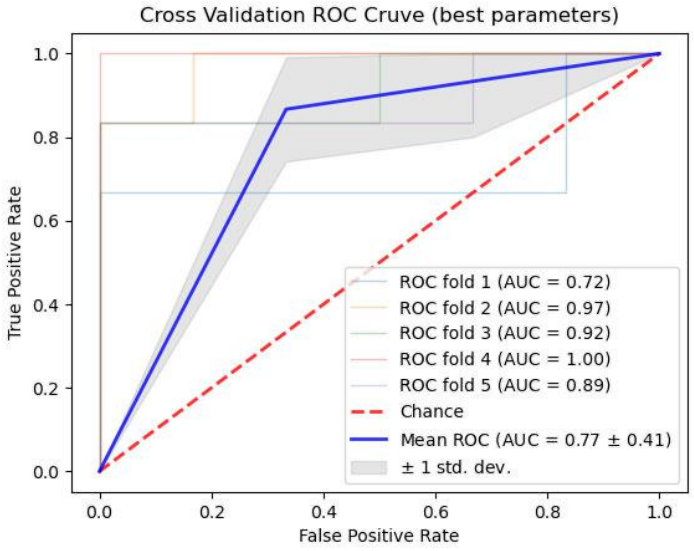

Figure 19. ROC report for Model #1 - experiment 2-L. Source: Author.

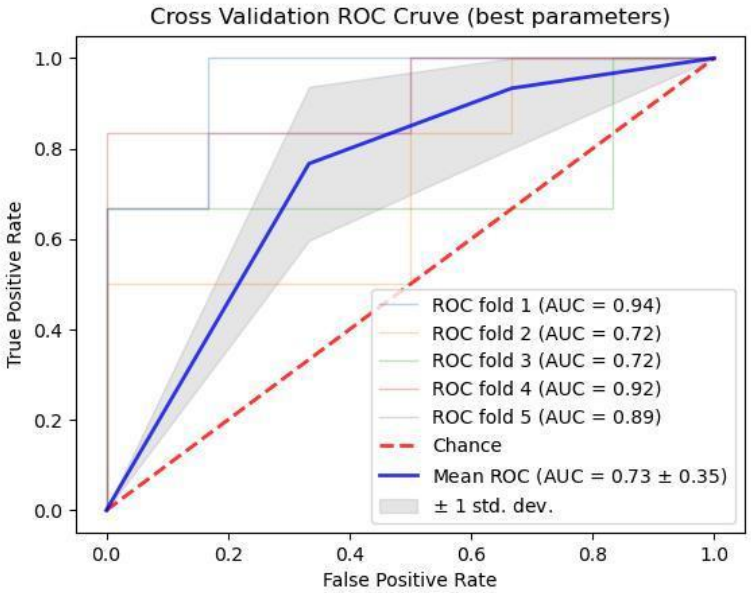

Figure 20. ROC report for Model #1 - experiment 2-R. Source: Author.

6.1.3. Experiment 3

According to Table 6, when considering the mean of both sides for each class, the results improved by an average of 4.7% and 2.8% with respect to experiments 1, and 2. Moreover, the stability of the ACC results improved, with a std of 6.2%. Furthermore, it is evident that this model is more suitable to detect patients than those that consider left and right sides separately. It is evident that the information from both sides provides more information for the classifier.

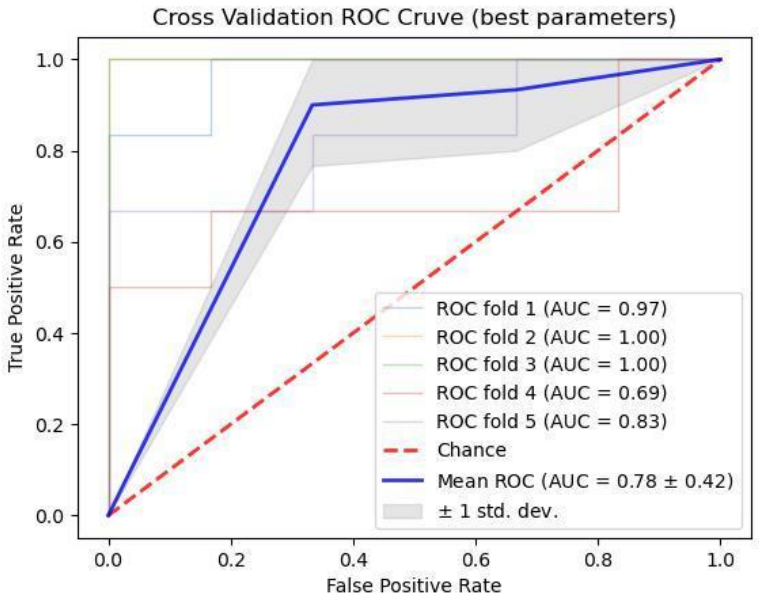

Figure 21. ROC report for Model #1 - experiment 3. Source: Author.

6.1.4. Experiment 4

The signals from the mobile phone, and the eGait system are differentiable, according to the results in Table 6. Despite the representations from the AEs, such signals can be differentiated with high accuracy. The big issue with these experiments is that the origin of both signals is different. Both models exhibit a similar ROC report that the previous ones (Figures 22 & 23).

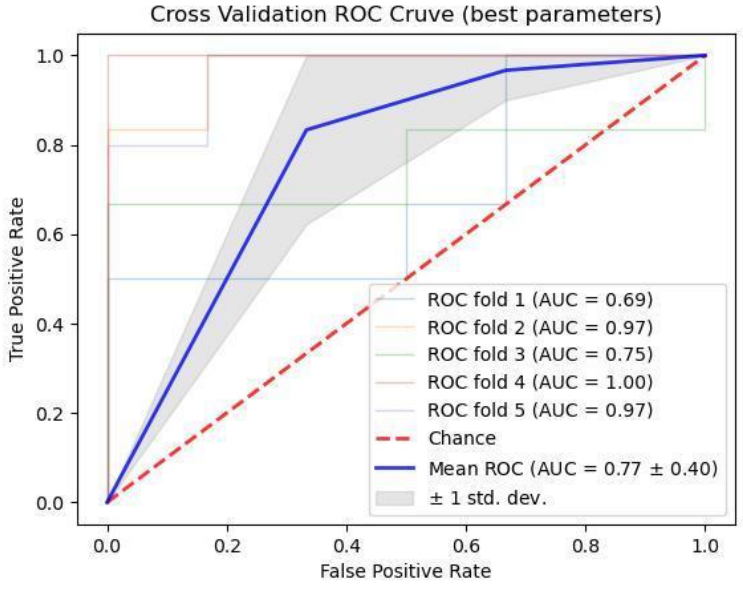

Figure 22. ROC report for Model #2 - experiment 4-L. Source: Author.

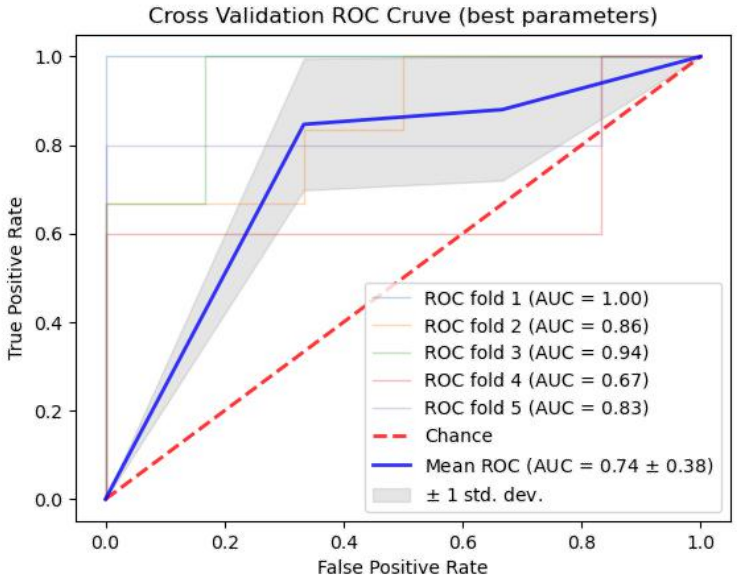

Figure 23. ROC report for Model #2 - experiment 4-R. Source: Author.

6.2. Model #2 (128-dimensional embeddings)

Experiment	ACC (%)	SEN	SPEC	UAR	Best Model
1	50.00 ± 9.00	0.39 ± 0,21	0.52 ± 0.23	0.46	C=1 gamma=0.1 kernel=RBF
2	78.30 ± 10.000	0.93 ± 0.13	0.93 ± 0.08	0.93	C=1, gamma=0.1, kernel= RBF
	78.30 ± 6.71	0.96 ± 0.06	0.96 ± 0.06	0.96	C=1, kernel=Linear
3	80.00 ± 8.50	0.95 ± 0.10	0.97 ± 0.06	0,96	C=1, kernel=Linear

Table 7. Results for Model #2. 128-dimensional embeddings.

6.2.1. Experiment 1

Compared to the results in section 6.1.1, the results seem to improve. However, the same problem is present. When considering the mean, std, kurtosis and skewness, the results improve to **ACC** 74.1 ± (9.9), **SEN** 0.66 ± 0.50, **SPEC** 0.61 ± 1.26, and **UAR** 0.63. There is a 24.1% average improvement in ACC, 40% in sensitivity, 19% in specificity, and a considerable increase in the UAR with an average of 17%. Figure 24 Illustrates the result.

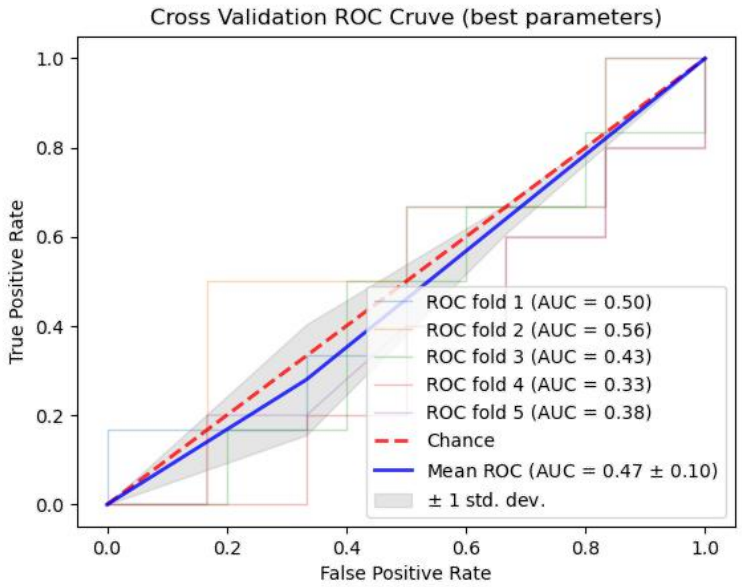

Figure 24. ROC report Model #2 - experiment 1. Source: Author.

6.2.2. Experiment 2

The results indicate that there is not a considerable difference in the ACC scores, SEN, SPEC and UAR, indicating that there is a possibility in which the 64-D embeddings provide sufficient information to represent both patients and HCs for the classification tasks. Figures 25 and 26 show the ROC curve report for this case.

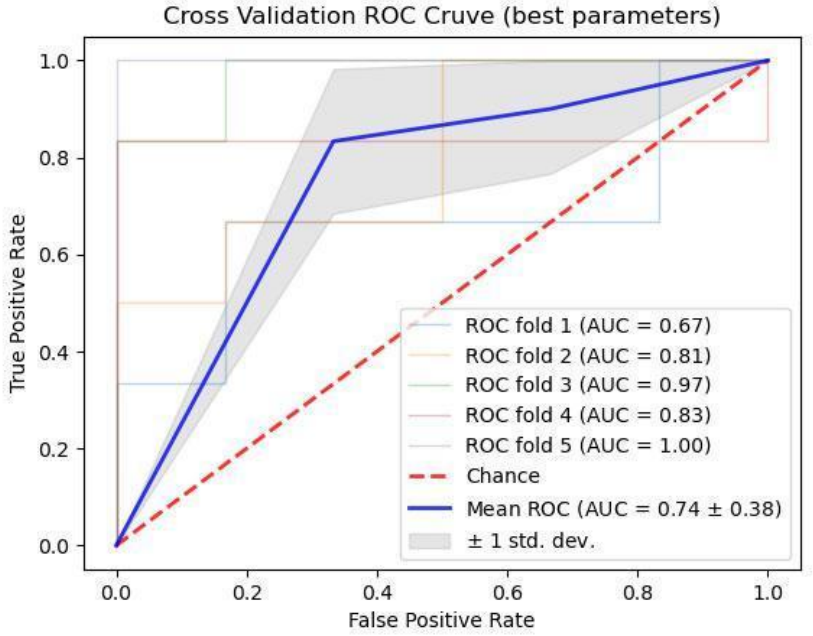

Figure 25. ROC report Model #2 - experiment 2-L. Source: Author.

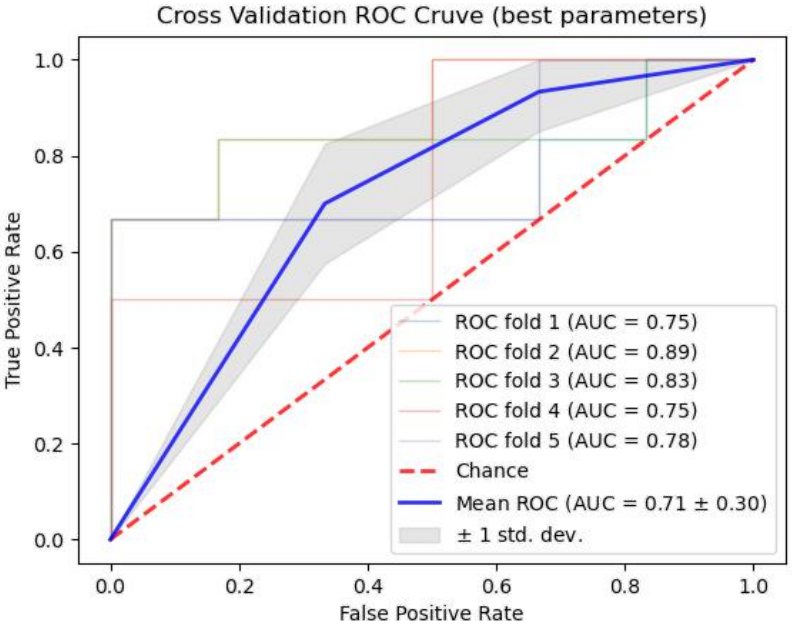

Figure 26. ROC report Model #2 - experiment 2-R. Source: Author.

6.2.3. Experiment 3

Similarly, this experiment does not show important improvements in the classification metrics. One possible explanation for such results is that the smaller bottleneck of Model #1 contains more meaningful information of the subjects. Figure 27 illustrates the result for this section.

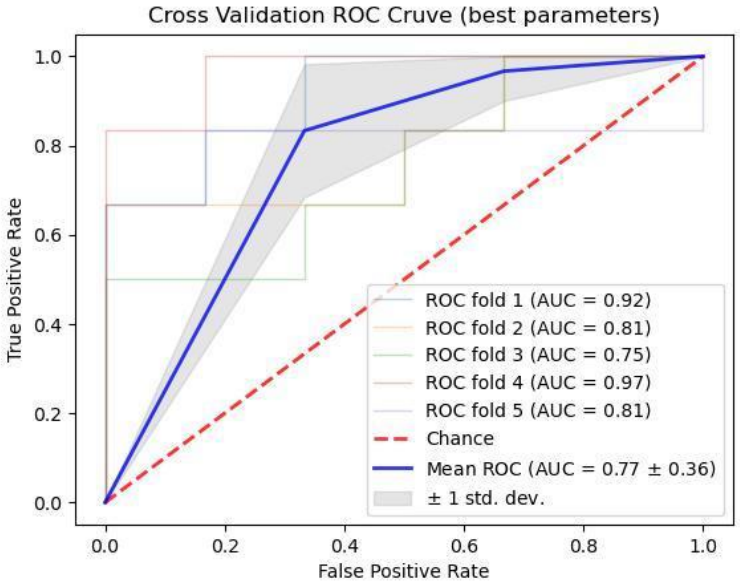

Figure 27. ROC report Model #2 - experiment 3. Source: Author.

6.3. Model #3 (256-dimensional embeddings)

Experiment	ACC (%)	SEN	SPEC	UAR	Best Model
1	73.90 ± 15.00	0.73 ± 0.16	0.68 ± 0.16	0.71	C=0.01, gamma=0.001 kernel=RBF
2	85.00 ± 18.600	0.87 ± 0.19	0.90 ± 0.13	0.88	C=0.001, gamma=0.1, kernel= RBF
	78.30 ± 14.50	0.86 ± 0.18	0.80 ± 0.26	0.83	C=1, gamma=0.001, kernel=RBF
3	81.70 ± 9.70	0.83 ± 0.09	0.83 ± 0.10	0,83	C=1, gamma= 0.001, kernel=RBF

Table 8. Results for Model #3. 256-dimensional embeddings.

As specified by Table 8, there is not a substantial difference in the results using higher dimensions in the bottleneck of the CAE, with the exception of experiment 1. In experiments 1 and 2, all metrics improved when considering more features. However, in this experiment, only the mean+std were considered. It is evident that the signals from the smartphones are more difficult to classify, given the recording constraints of the Android devices. Such constraints include different accelerometer manufacturers, and variable sampling frequency, depending on resource availability. Since the devices used for recording these signals had variability, the quality of the data is different when compared to the eGait signals. Further research is needed with this source, to determine more approaches to analyse the impact of these constraints in the classification metrics. Figure 28 shows the ROC report for this experiment.

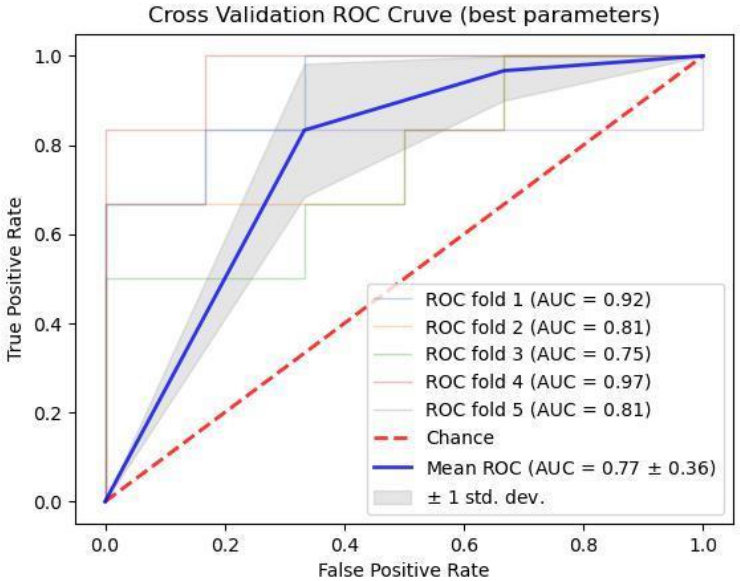

Figure 28. ROC report for Exp 3:1. Source: Author.

6.4. Non-randomized channels

Tables 9-11 Illustrate the results of training the CAENC with non-randomized channels.

Experiment	ACC (%)	SEN	SPEC	UAR	Best Model
1	46.20 ± 11.31	0.15 ± 0,31	0.52 ± 0.11	0.34	C=0.001, gamma=0.001, kernel= RBF
2	81.00 ± 9.31	0.92 ± 0.01	0.90 ± 0.10	0.90	C=1, gamma=0.1, kernel= RBF
	85.10 ± 13.31	0.83 ± 0.19	0.97 ± 0.12	0.90	C=1, gamma=1, kernel=RBF
3	83.10 ± 8.00	0.93 ± 0.13	0.93 ± 0.02	0,93	C=1, gamma=1 kernel= RBF
4	79.70 ± 13.26	0.88 ± 0.12	0.84 ± 0.30	0.86	C=1, kernel=Linear
	84.90 ± 6.21	0.99 ± 0.06	0.98 ± 0.12	0.99	C=1, kernel=Linear

Table 9. Results for Model #1. Non-randomized.

Experiment	ACC (%)	SEN	SPEC	UAR	Best Model
1	51.78 ± 6.00	0.42 ± 0,14	0.57 ± 0.39	0.49	C=1 gamma=0.1 kernel=RBF
2	76.35 ± 12.25	0.90 ± 0.13	0.93 ± 0.08	0.92	C=1, gamma=0.1, kernel= RBF
	78.91 ± 8.41	0.96 ± 0.06	0.96 ± 0.06	0.96	C=1, kernel=Linear
3	82.02 ± 8.50	0.93 ± 0.27	0.95 ± 0.13	0,94	C=1, kernel=Linear

Table 10. Results for Model #2. Non-randomized.

Experiment	ACC (%)	SEN	SPEC	UAR	Best Model
1	75.10 ± 12.23	0.70 ± 0.16	0.69 ± 0.16	0.69	C=0.01, gamma=0.001 kernel=RBF
2	83.00 ± 18.600	0.76 ± 0.12	0.85 ± 0.10	0.81	C=0.001, gamma=0.1, kernel= RBF
	78.30 ± 14.50	0.79 ± 0.01	0.80 ± 0.10	0.80	C=1, gamma=0.001, kernel=RBF
3	83.10 ± 4.72	0.80 ± 0.09	0.80 ± 0.30	0,80	C=1, gamma= 0.001, kernel=RBF

Table 11. Results for Model #3. Non-randomized.

Randomizing the channels does not seem to affect the classification process, such a behavior is expected, as the information from all channels is compressed inside the bottleneck, and after a few iterations, the CAE will learn how to parse the information despite the channel of origin. The experiment conducted using randomized channels is the one suitable for future tasks, as it follows the principle of generalization.

There is enough information to conclude the results of this work, therefore, it is not necessary to present any more figures related with the ROC curve. Model #1 provides high ACC, SEN, SPEC, and UAR values overall, with the smallest bottleneck. In order to improve the detection rates in the Apkinson signals, more features such as the kurtosis and skewness should be considered. Considering this case. There are not any considerable improvements in the proposed metrics when using Models #2 and #3. Following the results in Tables 6-8. Most of the stds in the tables indicate that there is a small variability inside the hyperparameters from the stratified 5-Fold CVs. Further research is needed to verify if the inclusion of more features may lead to reduce such variations and improve the classification results. There is a huge gap between the signals from the eGait system and the ones obtained via Apkinson. Experiments with detailed information about which pocket the smartphone is placed in at all times are needed, in order for experiments including Apkinson vs eGait to really become meaningful. The results show that efficient representations of gait signals could be used in the future for clinical applications using Convolutional Auto-Encoders. Additionally, a possible investigation line could be followed using the results of this work, to better understand the possible uses of Smartphone based signals in PD classification and to predict the neurological state of the patients.

7. Conclusions

This study used Convolutional Auto-Encoders to extract features from gait signals of Parkinson's Disease patients and Healthy subjects. The signals were extracted from the Apkinson smartphone app and the eGait system. A quadratic interpolation process was used to equalize the sampling frequencies from both sources to 50Hz. A total of 38 PD patients and 38 HCs were used to train 6 models, 3 using randomized channels, and 3 using non-randomized channels. The inputs of the models were 3-channel, 125-dimensional windows extracted from the gait signals. The testing dataset is conformed of 30 PDs and 30 HCs, all subjects had Apkinson and eGait data. The testing dataset comprised 30 PDs and 30 HCs. Each set of Auto-Encoders used

64, 128, and 256-dimensional bottlenecks respectively to learn efficient representations from the signals. The feature sets were composed of the embedding vector from each recording. The early fusion of the mean and standard deviation from the feature vectors was considered for the classification task.

The goal was to model the gait impairments of the patients, in order to classify them vs HCs. A stratified 5-Fold Cross-Validation strategy was conducted, using a Support Vector Machine. The hyperparameter optimization was performed according to the accuracy on the development set. For each experiment, the Accuracy, Sensitivity, Specificity, Unweighted Average Recall and ROC Curve summary were reported. The results indicate that the 64-dimensional bottleneck model provides enough information to classify the PDs. The best result overall using such a model indicates an average Accuracy of 85%, and an Unweighted Average Recall of up to 93%. This result was obtained using information from both sides of the eGait system. Additionally, the signals from Apkinson needed more statistical functionals in order to differentiate PD patients vs. HC subjects. When considering the early fusion of the mean, standard deviation, skewness, and kurtosis, this classification process reaches average accuracy values 71.3%, and UAR values of 0.67. The 128 and 256- dimensional models did not provide significant improvements. When using the latter. Average accuracy values of 73.9% and UAR of 0.71 were obtained for the classification process using only Apkinson signals.

Experiments comparing non-randomized vs. randomized channels indicated that there are not substantial differences, as the resulting loss values are similar to the ones in the previous models and the classification is not affected by such a change. Further research is required with the aim of investigating into the analysis of the Apkinson signals using these kinds of methods.

8. References

[1] Balestrino, R. Schapira, A. H. V. (1999). Parkinson's disease. *BMJ*, (Vol. 318, pp. 311– 314),

[2] Perez-Lloret, S. Negre-Pages, L. P. (2014, July). Prevalence, Determinants, and Effect on Quality of Life of Freezing of Gait in Parkinson Disease. *JAMA*, (Vol. 71, pp. 884–890).

[3] Morris, M. E., Martin C. L., & Schenkman, M. L. Striding Out with Parkinson Disease: Evidence-Based Physical Therapy for Gait Disorders. *Physical Therapy (*Vol. 90, pp. 280–288).

[4] Behrman, A. L., Cauraugh, J. H., & Light, K. E. (2000). Practice as an intervention to improve speeded motor performance and motor learning in Parkinson's disease. *Journal of the Neurological Sciences* (Vol. 174, pp. 127–136).

[5] Goutham, R., Fisch L., Srinivasan, S. (2003, Jan). Does This Patient Have Parkinson Disease?. *JAMA*, (Vol. 289, pp. 347–353).

[6] Dickson, D. W., Braak, H., Duda, J. E. (2009). Neuropatho-logical assessment of parkinson's disease: refining the diagnostic criteria. *The Lancet Neurology*. (Vol. 8, pp. 1150–1157).

[7] Goetz, C. G. Fahn, S., Martinez-Martin, P. Poewe, W. (2007). Movement disorder society-sponsored revision of the unified Parkinson's disease rating scale (MDS-UPDRS): Process, format, and clinimetric testing plan. *Movement Disorders*, (Vol. 22, pp. 41–47).

[8] Artusi, C. A., Mishra, M., & Latimer, P. (2017, Nov). Integration of technology-based outcome measures in clinical trials of Parkinson and other neurodegenerative diseases. *Proceedings of XXII World Congress of the International Association of Parkinsonism and Related Disorders*. (Vol. 46). Ho Chi Minh City, Vietnam.

[9] Merola, A. Sturchio, A. Hacker, S. (2018, Sept). Technology-based assessment of motor and non-motor phenomena in Parkinson disease. *Expert Review of Neurotherapeutics.* (Vol. 18).

[10] Orozco-Arroyave, J. R., Vásquez-Correa, J. C., Klumpp, P. (2020). Apkinson: the smartphone application for telemoni-toring parkinson's patients through speech, gait and hands movement. *Neurodegenerative Disease Management*. (Vol. 10, pp. 137–157).

[11] *Vasquez-Correa, J. C., Arias-Vergara T. Klumpp, P., et al. (*2021*)* End-2-End Modeling Of Speech And Gait From Patients With Parkinson's Disease: Comparison Between High Quality vs. Smartphone Data. *ICASSP 2021*. (pp. 7298-7302).

[12] Perez-Toro, P., Vásquez-Correa J. C., Arias-Vergara, T. Noeth, E. (2020, Jun). Nonlinear dynamics and poincaré sections to model gait impairments in different stages of parkinson's disease. *NonlinearDynamics*, (Vol. 100, pp. 3253-3276).

[13] Camps, J. Sama, A., Martín, M. et al. (2017, Oct). Deep learning for freezing of gait detection in parkinson's disease patients in their homes using a waist-worn inertial measurement unit. *Knowledge-Based Systems.* (Vol. 139, pp. 119-131).

[14] Sherstinsky, A. (2020, Mar). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*. (Vol. 404, pp. 132-306).

[15] Balaji, E. Brindha, D., Vikrama, R. et al. (2021). Automatic and non-invasive Parkinson's disease diagnosis and severity rating using LSTM network. *Applied Soft Computing*. (Vol. 108).

[16] Khokhlova, M., Migniot C., Morozov A., et al. (2019). Normal and pathological gait classification LSTM model. *Artificial Intelligence in Medicine*, (Vol. 94, pp. 54-66).

[17] Goetz, C., Tilley, B., Shaftman, S. et al. (2008). Movement Disorder Society-sponsored revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and clinimetric testing results. *Movement Disorders*, (Vol. 23, pp. 2129-2170). 2008.

[18] Djurić-Jovičić, M., Petrović, I., Ječmenica-Lukić, M., Radovanović, S. et al. (2016, Aug). Finger tapping analysis in patients with Parkinson's disease and atypical parkinsonism. *J Clin Neurosci*. (Vol. 30, pp. 49-55).

[19] Süli, E., Mayers, D. (2012, Jun). An Introduction to Numerical Analysis. *Cambridge University Press*.

[20] JAKKULA, V. (2006) Tutorial on support vector machine (SVM). *School of EECS*, 2006. (Vol. 37).Washington State University.

[21] BROWNE M. W. (2000)., Cross-validation methods. *Journal of mathematical psychology*. (Vol. 44, pp. 108-132).

[22] Hiromu, Y., Shinozaki, S., Nishimura, R. et al. (2018, Mar). Malware Analysis of Imaged Binary Samples by Convolutional Neural Network with Attention Mechanism. *Conference: The 8th ACM Conference on Data and Application Security and Privacy*. (pp 127-134).

[23] Fine, T. L. (1999). Feedforward Neural Network Methodology. *Information Science and Statistics*, Springer New York.

[24] Atienza, R. (2018). Advanced Deep Learning with Keras: Apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy gradients, and more. *Packt Publishing Ltd*.

[25] Michelucci, U. Advanced Applied Deep Learning, *Apress Berkeley*, CA.