An Evolutionary Higher-Order Extreme Learning Machine Method for Parkinson's Disease Classification

Vasileios Christou[a,*], Alexandros T. Tzallas[a], Georgios Tsoumanis[a], Markos G. Tsipouras[b], Dimitrios Dimopoulos[c], Dimitrios Varvarousis[c], Avraam Ploumis[c], Nikolaos Giannakeas[a]

[a]*Department of Informatics and Telecommunications, University of Ioannina, Arta, Greece*
[b]*Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, Greece*
[c]*Department of Physical Medicine and Rehabilitation, University of Ioannina, S. Niarchos Ave, 45110, Ioannina, Greece*

[*]Corresponding Author.

Email addresses: bchristou1@gmail.com (Vasileios Christou), tzallas@uoi.gr (Alexandros T. Tzallas), gtsoum@uoi.gr (Georgios Tsoumanis), mtsipouras@uowm.gr (Markos G. Tsipouras), ddimop@uoi.gr (Dimitrios Dimopoulos), dimvarvar@gmail.com (Dimitrios Varvarousis), aploumis@uoi.gr (Avraam Ploumis), giannakeas@uoi.gr (Nikolaos Giannakeas)

Abstract

Parkinson's disease is a neurodegenerative disease that primarily impacts the patient's motor system. A large percentage of the patients at the early stages of the disease develop vocal disorders. This finding is utilized from telediagnosis systems aimed at early disease detection. The current study uses a dataset containing voice recordings from 252 subjects. These recordings have undergone a feature extraction, a pre-processing, and a feature selection procedure before being sent to a neural network hybrid algorithm to classify them into two categories (normal and abnormal). The hybrid algorithm creates and trains, using higher order extreme learning machine (HO-ELM) algorithm, a set of higher-order multi-cube unit (MCU) single-layer neural networks (SLNNs). The ELM algorithm gained popularity mainly due to its simplicity and fast training speed. ELM is used with traditional low-order neuron types where each input vector element is multiplied with a corresponding weight vector value. This neuron type is restricted to solving linear separable problems. Non-linear separable ones would require using a neural network or more advanced neuron types like the higher-order unit. The latter has a weight vector containing more entries than the input vector. A higher-order unit type is the MCU which utilizes a series of hyper-cubes where each site of these cubes corresponds to a weight value. These trained MCU SLNNs are evolved using a modified genetic algorithm (GA), and the most optimal one is selected for the classification process. The GA is self-adaptive and can determine the optimal number and size of each sub-cube forming every MCU of the network (hidden and output layers). Also, it can tune the hidden layer weights. The algorithm's accuracy was tested with 15 machine-learning methods and achieved the highest accuracy percentage. Finally, the statistical significance of the experimental results was tested using the Wilcoxon signed-rank test.

Keywords:
extreme learning machine, higher-order neuron, genetic algorithm, multi-cube neuron, Parkinson's disease

1. Introduction

Parkinson's disease (PD) is one of the most common neurodegenerative diseases affecting many patients' speech, posture, and gait. It was initially described by Dr. James Parkinson in 1819 in his work "An Essay on the Shaking Palsy", which was reprinted in 2002 (Parkinson, 2002). PD causes nerve cell degeneration in a small brain area named substantia nigra. The affected cells lose dopamine, a chemical substance aiding the communication of messages or signals from the brain to various human body regions. The patient faces movement issues when low dopamine levels compromise the brain's ability to transmit messages. The nerve cell degeneration causes tremors and stiffness in the patient's limbs as well as issues with gait and balance. Besides pathological problems, the patient might face psychological issues like anxiety and depression. PD is a long-lasting disease affecting approximately 10 million people worldwide, significantly reducing the patient's quality of life at later stages (Marie, 2020; Factor & Weiner, 2007).

Most patients with PD are elderly (over 60 years old) (Van Den Eeden et al., 2003) and have an extended life span, with their treatment usually requiring pharmacological or surgical means. Due to the nature of this disease, remote diagnosing and monitoring systems can be a powerful tool for the healthcare system. Such systems can detect early stages of the disease and reduce the number of patient visits to the hospitals, which in turn reduces the workload of healthcare specialists (Van Den Eeden et al., 2003; Erdogdu Sakar et al., 2017; Little et al., 2008; Sakar et al., 2019).

PD remote diagnosis and monitoring systems estimate the progression of the disease by using non-invasive methods. One critical symptom shown in a significantly large percentage of patients is vocal issues. Due to this observation, many telemedicine systems focus on vocal disorders (Gürüler, 2017; Peker, 2016; Sakar et al., 2013). These systems utilize different speech signal processing algorithms for extracting features which are then sent as input into various machine learning algorithms for creating decision support systems (Sakar et al., 2019).

The current study utilizes the "Parkinson's disease classification" dataset (Sakar et al., 2018), which contains 753 extracted features from the voice recordings of 188 subjects having PD and 64 normal ones. The dataset contained one additional column containing each participant's identification (ID) number, which has been removed. These features underwent a feature selection process, and the 50 best ones have been selected using the minimum redundancy maximum relevance (MRMR) algorithm. The MRMR algorithm selects a sub-feature set containing the maximum correlation with the class (output) and the minimum correlation between the selected features (Ding & Peng, 2005). This subset is entered as input to the evolutionary higher-order extreme learning machine algorithm (EHO-ELM) proposed by Christou et al. (2023), which is responsible for the classification task of the subjects into two categories (PD and non-PD patients).

EHO-ELM is a hybrid approach combining the higher-order extreme learning machine algorithm (HO-ELM) with a modified self-adaptive version of a genetic algorithm (GA) (Holland, 1975). HO-ELM by Christou (2023) extends the traditional extreme learning machine (ELM) algorithm developed initially by Huang et al. (2004, 2006) for single-layer neural networks (SLNNs) to SLNNs containing the higher-order neurons proposed by Gurney (1989) in their hidden and output layers. ELM can train an SLNN by treating it as a system of linear equations where the hidden layer weights and thresholds are randomized, and the output layer weights are analytically calculated with the help of the Moore-Penrose pseudo-inverse (Huang et al., 2004, 2006).

The advantages of ELM over other learning methods include very fast training speed since it is not an iterative method like gradient-based methods and simplicity since it doesn't contain any user-defined parameters affecting the training process (e.g., the learning rate in gradient-based approaches). Moreover, it doesn't fall into local minima. Besides the advantages mentioned above, it contains some significant disadvantages. One of them is the random selection of the hidden layer weights and thresholds, which might lead to poor generalization performance of the network. Another disadvantage of ELM is that the original algorithm works with traditional semi-linear (low-order) neuron types restricted to linear separable problems. Semi-linear units multiply each neuron input with a corresponding weight, and they are added together with an optional threshold before being sent to a transfer (output) function, which produces the neuron's output. Gurney's cubic (higher-order) units solve this problem by mapping the inputs to a hyper-cube where each cube site corresponds to a weight. This way, an $n-$input neuron contains 2^n weights with $n \in \mathbb{N}$. Although these neuron types can solve non-linear separable problems, they face scaling issues since the number of weights increases exponentially. Due to this fact, Gurney proposed the multi-cube unit (MCU), which breaks down one hypercube into smaller sub-cubes having different dimensions. HO-ELM networks containing MCU in both their layers are termed MCU-ELM networks and are used in the creation of the EHO-ELM algorithm (Huang et al., 2004, 2006; Gurney, 1989; Christou et al., 2023).

The motivation behind the creation of EHO-ELM was twofold. The first reason was to solve the random initialization problem of the hidden weights and thresholds. The second reason was to find an optimal combination of sub-cube units for the hidden and output layer neurons. These issues were dealt with using a modified self-adaptive GA, which creates and trains a series of MCU SLNNs and evolves them at each algorithm iteration into SLNNs with better generalization performance. At the end of the evolution process, the algorithm selects the most optimal one for the classification process of the "Parkinson's disease classification" dataset. EHO-ELM retains ELM's simplicity by making all GA's parameters self-adaptive (Christou, 2023). EHO-ELM was utilized because the evolutionary process created more optimal SLNNs than ELM and MCU-ELM and overperformed 15 machine learning methods, as seen in the experimental part of this article.

The article is structured into seven main sections starting with the "Introduction", where the problem's description and motivation are presented along with a brief description of the proposed system's architecture. The following section is the "Literature review", which explores existing research methods that show similarities to the proposed approach. The third section is "Related work", divided into six sub-sections analyzing the structure of cubic and MCU units, traditional ELM algorithm, MCU-ELM, the architecture of a GA, and EHO-ELM. The fourth section explains the architecture of the proposed system, while the next section presents the experimental results, followed by a "Discussion" and "Conclusion" section.

2. Literature review

Many methods have been proposed for PD remote diagnosis and monitoring. Some of them utilize vocal data for this task. Narendra et al. (2021) proposed two architectures (a classical pipeline method and an end-to-end system) for automatic PD detection from voice data between healthy and PD patients. The classical pipeline method utilized baseline and glottal features to train a support vector machine (SVM) algorithm. The end-to-end system used raw speech and voice source waveforms and utilized two glottal inverse filtering methods (iterative adaptive inverse filtering and quasi-closed phase analysis). Moreover, it applied the zero-frequency filtering approach. The classification task was done with the help of a convolutional neural network (CNN) which utilized a multi-layer perceptron (MLP). Xue et al. (2023) proposed a local dynamic feature selection fusion approach for PD diagnosis and severity prediction. Initially, it applies the maximal information coefficient to remove features with low relevance. Then, it utilizes a feature selection stage based on self-organizing map network clustering to create the input vector for both the Gaussian process classifier, responsible for the diagnosis task, and the random forest (RF) regressor, responsible for the severity prediction task. Despotovic et al. (2020) created a PD diagnosis and progression estimation system from voice recordings. The data undergo a feature selection stage using the automatic relevance determination algorithm and are introduced as an input vector to a Gaussian process classifier, producing the system's output.

Numerous methods utilize neural networks for the classification task. García-Ordás et al. (2023) created a deep-learning model for detecting and estimating PD severity. It utilizes a mixed MLP with classification and regression capabilities trained using patients' voice recordings. The detection task involves distinguishing between two categories (severe and non-severe) while the disease's estimation uses the unified PD rating scale (UPDRS). The MLP adopted a trained auto-encoder for removing features containing non-relevant information, which would reduce its generalization performance. Asmae et al. (2020) utilized an artificial neural network (ANN) for PD detection, which contained two hidden layers. The ANN was trained using 22 acoustic features extracted from a data collection containing healthy and non-healthy vocal phonations. Shahid & Singh (2020) created a PD prediction progression system based on UPDRS, which applies principal component analysis to the dataset to solve its multicollinearity problems and reduce its dimensionality. The resulting data make the input vector to the proposed deep neural network (DNN) algorithm, and along with a tuned parameter norm penalty, the DNN evaluates PD's progression by predicting motor and total UPDRS scores. Wodzinski et al. (2019) used the residual network (ResNet) deep learning model for PD detection from voice recordings. The authors determined the audio recordings' spectrum and sent that information into the ResNet architecture. ResNet had been pre-trained on data from the ImageNet (Russakovsky et al., 2015) and Saarbrüken voice disorder (Alhussein & Muhammad, 2018) databases. Moreover, the data collection was augmented in the time domain to avoid overfitting. The system was tested on the PC-GITA (Orozco-Arroyave et al., 2014) database containing 50 healthy and 50 non-healthy subjects.

A significant number of existing works utilize ELM-based methods. Anter et al. (2023) created a regression model for monitoring PD progression using voice data. The proposed model contains a feature selection mechanism based on a binary version of the ant lion optimizer and a differential evolution ELM variant responsible for predicting UPDRS scores. Chen et al. (2016) proposed a hybrid system for early PD diagnosis using the kernel ELM (KELM) algorithm. The model applies the MRMR feature selection algorithm to the dataset before sending the data to a KELM classifier. Wang et al. (2017) developed a PD detection method based on a dataset containing voice recordings. It utilizes a binary version of the adaptive artificial bee colony algorithm for feature selection and a kernel-based weighted ELM variant for the classification task between healthy and non-healthy subjects. While a non-linear mapping of the kernel function is used to enhance the extent of linear separation, a weighted approach is taken to address the issue of unbalanced data. Moreover, a continuous version of the adaptive artificial bee colony algorithm is used for parameter tuning. Guatelli et al. (2023) utilized voice signal spectrograms and created a PD detection system that utilized the ELM classifier. Agarwal et al. (2016) developed a PD detection method that used speech data from 10 healthy and 20 non-healthy subjects. The data underwent a pre-processing stage where 26 features were extracted and formed the input vector to an ELM-trained neural network. Shahsavari et al. (2016) applied a hybrid particle swarm optimization algorithm as a feature extraction method in a PD data collection containing entries from patients and non-patients. Then, the extracted data were introduced as input to an ELM classifier. Das & Nanda (2023) combined cuckoo search with the voting ensemble weighted ELM algorithm. Weighted ELM is used in datasets containing imbalanced entries. The voting ensemble was adopted to circumvent the inconsistent results from weighted ELM due to the random initialization of input weights. The cuckoo search was used for feature extraction, while the voting ensemble weighted ELM approach classifies unknown data into two categories (healthy and non-healthy). The system was tested in the Oxford PD detection dataset (Little et al., 2008).

Several existing systems are based on ensemble-based approaches. Tunc et al. (2020) evaluated PD progression from patients' voice recordings. Their system adopted a two-phase architecture where the first phase involved a feature selection process using the Boruta algorithm. The second stage receives the selected features as an input vector to the extreme gradient boosting algorithm responsible for PD disease severity assessment. Jatoth et al. (2022) created a smart healthcare system based on fog technology. It detects and monitors PD based on Parkinson's telemonitoring voice dataset from the University of California Irvine (UCI) machine learning repository Dua & Graff (2017). The system utilizes an enhanced version of the synthetic minority oversampling technique (SMOTE) to deal with the dataset's class imbalance problem. Also, it uses the extreme gradient boosting algorithm for the classification task, which implements gradient-boosting decision trees (DTs) (Chen et al., 2015). Hireš et al. (2022) used a CNN ensemble for detecting PD in a collection of data containing voice recordings from 100 participants. The data pre-processing procedure involved transforming these recordings into the time-frequency domain and converting them into images. Then, the Gaussian blurring filter was applied to all figures to remove extreme outliers. These filtered images were introduced as input to the CNN ensemble, which classified the participants into two categories (healthy and non-healthy). Meghraoui et al. (2021) created a novel pre-processing method for PD detection based on data collections containing voice recordings, which applies relevance analysis on PD's disturbance, bio-mechanical, and neurological features. The proposed system's classification process involved utilizing three machine learning methods with RF achieving the best results.

EHO-ELM algorithm was proposed for the classification task in the Sakar et al. (2013) data collection based on earlier works by Christou (2023); Christou et al. (2023). They have shown that utilizing MCUs in all layers of an ELM-trained SLNN can significantly improve its generalization ability. Although the neural network-based approaches presented above get very good results in various datasets, they do not consider higher-order units, which can further increase the network's generalization ability.

A significant number of machine learning methods that can be used for classifying pre-processed voice recordings from PD and healthy subjects include the following ten neural network training algorithms based on the original back-propagation (BP) by Werbos (1974). The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is a member of the quasi-Newton family of methods for unconstrained numerical optimization, which determines the search direction using second-order derivatives. The quasi-Newton methods face two major issues. The first issue is the calculation complexity of the Hessian and its inverse, while the second is large memory requirements. BFGS solves the calculation complexity problem by approximating Hessian's inverse. The memory requirements are reduced by employing an update process where the approximated Hessian is updated after each stage in the optimization procedure. BFGS utilizes the second-order derivative update formula by Broyden (1970a,b), Fletcher (1970), Goldfarb (1970), and Shanno (1970) (Schweitzer et al., 2000). The Powell-Beale conjugate gradient (PBCG) method is a conjugate gradient (CG) variant containing an automatic restart mechanism that considers the objective function. CG is a suitable algorithm for minimizing functions with a large number of variables because it does not store any matrices. The convergence rate of CG is linear unless its iterative process is occasionally restarted (Powell, 1977). The Fletcher-Reeves CG (FRCG) is a quadratically convergent gradient approach for finding an unconstrained local minimum of a multivariate function utilizing the conjugate direction update formula by Fletcher-Reeves (Fletcher & Reeves, 1964; Scales, 1985). Similar to FRCG, the Polak-Ribiére CG (PRCG) is a CG variant that uses the conjugate direction update formula proposed by Polak-Ribiére (Polak & Ribiere, 1969). The scaled conjugate gradient (SCG) approach has a super-linear convergence rate, employs second-order neural network information, and requires $O(N)$ memory usage (N denotes the neural network weights number). SCG is at least an order of magnitude faster than traditional BP and requires no user-defined parameters (Møller, 1993). The steepest descent (SD) is one of the simplest function minimization methods but suffers from slow convergence (Cauchy et al., 1847; Meza, 2010). Gradient descent with momentum (GDM) enhances the original gradient descent (GD) algorithm by adding a momentum that permits the network to disregard minor features on the error surface. This way, GDM can bypass local minimums and converge faster (Polyak, 1964). Similarly, GD with momentum and adaptive learning rate (GDMALR) also enhances the original GD algorithm by adding momentum and an adaptive learning rate. One-step secant (OSS) is an approach between quasi-Newton and CG methods. It has fewer computational and memory requirements than BFGS, but it has a bit higher computational and memory requirements than CG-based algorithms. Its main advantage lies in calculating the new search direction, which does not require a matrix inverse calculation Battiti (1992). Resilient BP (RPROP) (Riedmiller & Braun, 1993) enhances GD by applying a local adaptation of the weight updates according to the error function's behavior.

Other non-BP-based approaches include the SVM algorithm, invented in the early 1990s as a non-linear solution for classification and regression problems. It initially maps the input data from a binary classification problem to a high-dimensional feature space. Then, with the help of a hyperplane, it can classify unknown data. The hyperplane is created by maximizing its distance from the nearest data points which belong to the first or second category (Cortes & Vapnik, 1995; Samui et al., 2017; Vapnik, 1998). DTs are hierarchical decision support models which utilize tree-like representations of options and their potential outcomes Von Winterfeldt & Edwards (1986). Finally, the online sequential ELM (OS-ELM) algorithm is a modified ELM version that works with sequential data (Huang et al., 2005).

3. Related work

The "Related work" section initially presents the cubic and MCU structures developed by Gurney (1989). The latter defines the neuron types used for both layers of the SLNNs that are evolved using the EHO-ELM method. Then, the ELM training algorithm and its extension for MCUs (MCU-ELM) are analyzed, followed by a presentation of the typical structure of a GA. MCU-ELM trains a series of MCU SLNNs, which are evolved utilizing the custom GA by the EHO-ELM method shown in the last sub-section of the "Related work". EHO-ELM automatically selects the SLNN with the best generalization performance, which contains an optimal combination of hidden layer weights and sub-cube units for all neurons.

3.1. The cubic unit structure

The main difference between the cubic units by Gurney (1989) compared to low-order ones is that they are more advanced structures containing a higher number of weights. Unlike low-order neurons, their weight assignment does not follow the $1-1$ rule (one weight per input). Instead, they follow the formula $w_{no}^{c} = 2^{n}, n \in \mathbb{N}$ with the c superscript denoting the cubic neuron type and n the inputs' number. Cubic neurons are regarded as site values in a hyper-cube where the number of inputs defines the hypercube's dimension. An orthographic projection of a tesseract denoting the weight assignment of a 4-input cubic neuron is depicted in Fig. 1.

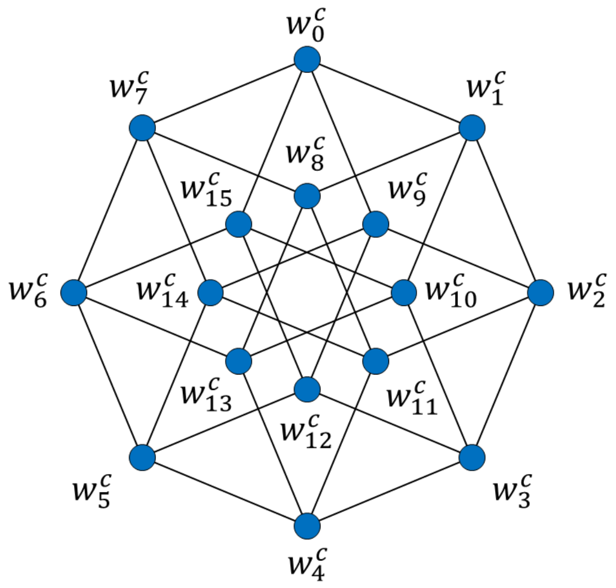

Figure 1: A tesseract. This figure shows the weight assignment of a 4-input neuron as site values on a tesseract.

Cubic neurons' weight assignment is defined by the probability function in (1). The n symbol denotes the number of inputs, while x_i defines the current input. The $\mu = \mu_1, \mu_2, \ldots, \mu_n$ inside product term $(1+\mu_i x_i)$ changes its plus arithmetic operator to minus and vise-versa. The procedure involves converting μ to a binary string format and assigning the plus or minus operators for each 1 or 0 found inside this binary string. Also, cubic neurons require normalization of the input vector, usually by dividing all input values by the highest absolute input value (Gurney, 1989).

$$P_\mu = \frac{1}{2^n} \prod_{i=1}^{n} (1 + \mu_i x_i) \tag{1}$$

The cubic unit activation defined in formula (2) multiplies every assigned weight with its probability calculated using equation (1). Afterward, it adds them together and calculates their normalized average using $\frac{1}{|w_{max^c}|}$ term. The normalization procedure involves dividing every weight (w_μ^c) with the highest absolute weight value $(|w_{max}^c|)$.

$$a^c = \frac{1}{|w^c_{max}|2^n} \sum_{\mu=0}^{2^n-1} w^c_\mu \prod_{i=1}^{n} (1 + \mu_i x_i) \tag{2}$$

The activation is then introduced as input to the transfer (activation) function g, which calculates the neuron's output value y. The procedure is visualized in Fig. 2 (Gurney, 1989).

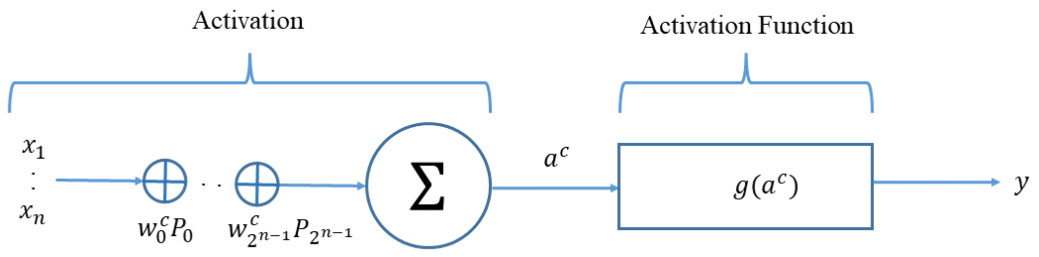

Figure 2: Cubic unit architecture. The input vector $[x_1, \ldots, x_n]$ is introduced to the cubic unit and is expanded using the probability function in (1). The probability function's outputs are multiplied element-wise with the cubic weights and added together. Then, they are averaged with the help of a normalized average creating the activation a_c. The activation is introduced as input to the transfer function g which produces the neuron's output value y (Christou et al., 2023).

3.2. The MCU structure

The number of weights in the cubic unit is defined by the formula $w^c_{no} = 2^n$ first introduced in section 3.1 where it is observed that the number of weights increases exponentially to a linear increase to the number of inputs (n). This fact makes them practically unusable for problems requiring many inputs due to the large increase in memory and computational requirements. To solve this issue, Gurney (1989) created the MCU, which divides a high-dimension hypercube into a series of sub-cubes with low dimensions. The MCU significantly reduces the computational and memory requirements compared to the cubic unit and scales well. The number of weights is defined using formula $w^{mc}_{no} = p_{no} = \sum_{j=1}^{q} 2^{d_j}, (j, d_j, q) \in \mathbb{N}$ where the mc superscript denotes the MCU type, and q is the number of sub-cubes. Vector $d = [d_1, d_2, \ldots, d_q]$ incorporates every sub-cube's dimension with current sub-cube dimension d_j deriving from d.

The MCU activation function is defined in equation (3) where the activation of each sub-cube is added together and normalized by dividing with the maximum absolute weight value $(\frac{1}{w^{mc}_{max}})$.

$$a^{mc} = \frac{1}{|w^{mc}_{max}|} \sum_{j=1}^{q} \frac{1}{2^{d_j}} \sum_{\mu=0}^{2^{d_j}-1} w^{mc}_{\mu} \prod_{i=1}^{d_j} (1 + \mu_i x_i) \tag{3}$$

The MCU activation is then introduced as input to the transfer (activation) function g, which calculates the neuron's output value y. The procedure is visualized in Fig. 3 (Gurney, 1989).

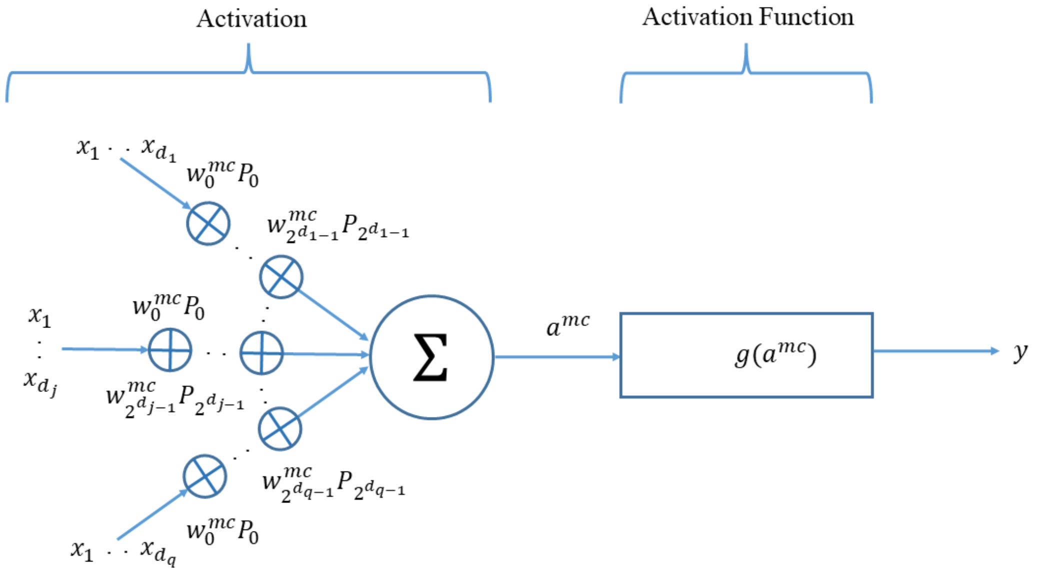
Activation
Activation Function
$x_1 \cdot \cdot x_{d_1}$
$w_0^{mc} P_0$
$w^{mc}_{2^{d_1-1}} P_{2^{d_1-1}}$
x_1
x_{d_j}
$w_0^{mc} P_0$
$w^{mc}_{2^{d_j-1}} P_{2^{d_j-1}}$
Σ
a^{mc}
$g(a^{mc})$
y
$w^{mc}_{2^{d_q-1}} P_{2^{d_q-1}}$
$w_0^{mc} P_0$
$x_1 \cdot \cdot x_{d_q}$

Figure 3: MCU unit architecture. The input vector $[x_1, \ldots, x_n]$ is introduced to the MCU unit, which assigns them to several sub-cubes. The input vector in each sub-cube is expanded using the probability function in (1). The probability function's outputs are multiplied element-wise with the sub-cube's weights, added together, and averaged, creating each sub-cube's activation, which in turn are added together and normalized by dividing with the maximum absolute weight value from all sub-cubes. The MCU activation is introduced as input to the transfer function g, which produces the neuron's output value y (Christou et al., 2023).

3.3. The ELM algorithm

The main difference of the ELM training algorithm is that it does not follow an iterative approach like BP-based methods and can work with non-differentiable transfer functions Huang et al. (2004, 2006). Moreover, Huang et al. (2006) utilized an incremental constructive approach to prove that SLNNS can have universal approximation capability by randomizing their hidden layer weights and thresholds and calculating their output layer weights. An ELM trainable SLNN has the mathematical model defined in formula 4

$$\begin{bmatrix} g\left(\begin{bmatrix} x_{1,1} \\ \vdots \\ x_{1,n} \end{bmatrix}^T \begin{bmatrix} w^l_{1,1} \\ \vdots \\ w^l_{n,1} \end{bmatrix} + \theta_1\right) & \dots & g\left(\begin{bmatrix} x_{1,1} \\ \vdots \\ x_{1,n} \end{bmatrix}^T \begin{bmatrix} w^l_{1,h} \\ \vdots \\ w^l_{n,h} \end{bmatrix} + \theta_h\right) \\ \vdots & \dots & \vdots \\ g\left(\begin{bmatrix} x_{N,1} \\ \vdots \\ x_{N,n} \end{bmatrix}^T \begin{bmatrix} w^l_{1,1} \\ \vdots \\ w^l_{n,1} \end{bmatrix} + \theta_1\right) & \dots & g\left(\begin{bmatrix} x_{N,1} \\ \vdots \\ x_{N,n} \end{bmatrix}^T \begin{bmatrix} w^l_{1,h} \\ \vdots \\ w^l_{n,h} \end{bmatrix} + \theta_h\right) \end{bmatrix}_{N\times h} \cdot$$
$$\begin{bmatrix} \beta^l_{1,1} & \dots & \beta^l_{1,m} \\ \vdots & \dots & \vdots \\ \beta^l_{h,1} & \dots & \beta^l_{h,m} \end{bmatrix}_{h\times m} = \begin{bmatrix} t_{1,1} & \dots & t_{1,m} \\ \vdots & \dots & \vdots \\ t_{N,1} & \dots & t_{N,m} \end{bmatrix}_{N\times m} \tag{4}$$

The activation function is declared with the symbol g while $x = \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \dots & \vdots \\ x_{N,1} & \dots & x_{N,n} \end{bmatrix}_{N\times n} \in \mathbb{R}^{N\times n}$ is the matrix containing the input data, $w^l = \begin{bmatrix} w^l_{1,1} & \dots & w^l_{1,h} \\ \vdots & \dots & \vdots \\ w^l_{n,1} & \dots & w^l_{n,h} \end{bmatrix}_{n\times h} \in \mathbb{R}^{n\times h}$ is the matrix containing the hidden layer weights and $\theta = [\theta_1 \dots \theta_h] \in \mathbb{R}^h$ is the hidden layer thresholds vector. The traditional low-order unit types are declared using the l superscript, n declares the number of neuron inputs, h is the number of hidden layer units, and N denotes the number of input patterns. The matrix $\beta^l = \begin{bmatrix} \beta^l_{1,1} & \dots & \beta^l_{1,m} \\ \vdots & \dots & \vdots \\ \beta^l_{h,1} & \dots & \beta^l_{h,m} \end{bmatrix}_{h\times m} \in \mathbb{R}^{h\times m}$ stores the output layer weights values where m declares the number of output neurons and the $T = \begin{bmatrix} t_{1,1} & \dots & t_{1,m} \\ \vdots & \dots & \vdots \\ t_{N,1} & \dots & t_{N,m} \end{bmatrix}_{N\times m}$ matrix stores SLNNs target values.

Algorithm 1 describes the structure of ELM, beginning with the randomization of the hidden layer weights and thresholds (lines 1 and 2). Then, it creates the hidden layer output matrix H, which stores the output values for each training entry to the network. The next step (line 4) contains the SLNN's target output matrix, while the last step calculates the output layer weights matrix using the multiplication of the Moore-Penrose pseudo-inverse of H with T.

Algorithm 1 : ELM

```
1: $w^l = \begin{bmatrix} w^l_{1,1} & \dots & w^l_{1,h} \\ \vdots & \dots & \vdots \\ w^l_{n,1} & \dots & w^l_{n,h} \end{bmatrix}_{n\times h}$
2: $\theta = [\theta_1, \dots, \theta_h]$
3: $H = \begin{bmatrix} g\left(\begin{bmatrix} x_{1,1} \\ \vdots \\ x_{1,n} \end{bmatrix}^T \begin{bmatrix} w^l_{1,1} \\ \vdots \\ w^l_{n,1} \end{bmatrix} + \theta_1\right) \dots g\left(\begin{bmatrix} x_{1,1} \\ \vdots \\ x_{1,n} \end{bmatrix}^T \begin{bmatrix} w^l_{1,h} \\ \vdots \\ w^l_{n,h} \end{bmatrix} + \theta_h\right) \\ \vdots \dots \vdots \\ g\left(\begin{bmatrix} x_{N,1} \\ \vdots \\ x_{N,n} \end{bmatrix}^T \begin{bmatrix} w^l_{1,1} \\ \vdots \\ w^l_{n,1} \end{bmatrix} + \theta_1\right) \dots g\left(\begin{bmatrix} x_{N,1} \\ \vdots \\ x_{N,n} \end{bmatrix}^T \begin{bmatrix} w^l_{1,h} \\ \vdots \\ w^l_{n,h} \end{bmatrix} + \theta_h\right) \end{bmatrix}_{N\times h}$
4: $T = \begin{bmatrix} t_{1,1} & \dots & t_{1,m} \\ \vdots & \dots & \vdots \\ t_{N,1} & \dots & t_{N,m} \end{bmatrix}_{N\times m}$
5: $\beta^l = H^\dagger T$
```

3.4. The MCU-ELM algorithm

MCU-ELM extends the traditional ELM algorithm to SLNNs having MCUs in both their layers (hidden and output). It was first introduced by Christou (2023), and its mathematical model is defined in equation 5. The P symbol defines MCU's probability function in this model, while g is the activation function. The MCU activations are stored in matrix $a^{mc} \in \mathbb{R}^{N\times h} =$

$$\begin{bmatrix} a^{mc}_{1,1}\left(\begin{bmatrix} x_{1,1}\\ \vdots \\ x_{1,n}\end{bmatrix}^T, \begin{bmatrix} w^{mc}_{0,1}\\ \vdots \\ w^{mc}_{p_h-1,1}\end{bmatrix}\right) & \dots & a^{mc}_{1,h}\left(\begin{bmatrix} x_{1,1}\\ \vdots \\ x_{1,n}\end{bmatrix}^T, \begin{bmatrix} w^{mc}_{0,h}\\ \vdots \\ w^{mc}_{p_h-1,h}\end{bmatrix}\right) \\ \vdots & \dots & \vdots \\ a^{mc}_{N,1}\left(\begin{bmatrix} x_{N,1}\\ \vdots \\ x_{N,n}\end{bmatrix}^T, \begin{bmatrix} w^{mc}_{0,1}\\ \vdots \\ w^{mc}_{p_h-1,1}\end{bmatrix}\right) & \dots & a^{mc}_{N,h}\left(\begin{bmatrix} x_{1,1}\\ \vdots \\ x_{1,n}\end{bmatrix}^T, \begin{bmatrix} w^{mc}_{0,h}\\ \vdots \\ w^{mc}_{p_h-1,h}\end{bmatrix}\right) \end{bmatrix}_{N\times h}.$$

In this matrix, $x = \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \dots & \vdots \\ x_{N,1} & \dots & x_{N,n} \end{bmatrix}_{N\times n} \in \mathbb{R}^{N\times n}$ defines the matrix containing all inputs while $w^{mc} = \begin{bmatrix} w^{mc}_{0,1} & \dots & w^{mc}_{0,h} \\ \vdots & \dots & \vdots \\ w^{mc}_{p_h-1,1} & \dots & w^{mc}_{p_h-1,h} \end{bmatrix}_{p_h\times h} \in \mathbb{R}^{p_h\times h}$ contains the hidden layer weights. The MCU types are declared using the mc superscript, n declares the number of neuron inputs, h is the number of hidden layer units, N denotes the number of input patterns, p_h denotes the number of hidden layer weights, while p_o is the output layer weights number. The matrix $\beta^{mc} = \begin{bmatrix} \beta^{mc}_{0,1} & \dots & \beta^{mc}_{0,m} \\ \vdots & \dots & \vdots \\ \beta^{mc}_{p_o-1,1} & \dots & \beta^{mc}_{p_o-1,m} \end{bmatrix}_{p_o\times m} \in \mathbb{R}^{p_o\times m}$ stores the output layer weights where m declares the output units number and matrix $T = \begin{bmatrix} t_{1,1} & \dots & t_{1,m} \\ \vdots & \dots & \vdots \\ t_{N,1} & \dots & t_{N,m} \end{bmatrix}_{N\times m} \in \mathbb{R}^{N\times m}$ stores the target values.

$$\begin{bmatrix} P_{1,0}\left(g(a^{mc}_{1,1}),\dots,g(a^{mc}_{1,h})\right) & \dots & P_{1,p_o-1}\left(g(a^{mc}_{1,1}),\dots,g(a^{mc}_{1,h})\right) \\ \vdots & \dots & \vdots \\ P_{N,0}\left(g(a^{mc}_{N,1}),\dots,g(a^{mc}_{N,h})\right) & \dots & P_{N,p_o-1}\left(g(a^{mc}_{N,1}),\dots,g(a^{mc}_{N,h})\right) \end{bmatrix}_{N\times p_o} \cdot \begin{bmatrix} \beta^{mc}_{0,1} & \dots & \beta^{mc}_{0,m} \\ \vdots & \dots & \vdots \\ \beta^{mc}_{p_o-1,1} & \dots & \beta^{mc}_{p_o-1,m} \end{bmatrix}_{p_o\times m} = \begin{bmatrix} t_{1,1} & \dots & t_{1,m} \\ \vdots & \dots & \vdots \\ t_{N,1} & \dots & t_{N,m} \end{bmatrix}_{N\times m} \tag{5}$$

Algorithm 2 shows the MCU-ELM structure, starting with randomizing the hidden layer weights (line 1). Afterward, it creates the hidden layer output matrix H, which stores the output values for each training entry to the network. Line 4 shows the SLNN's target output matrix, and the fourth step calculates the output layer weights matrix using the multiplication of the Moore-Penrose pseudo-inverse of H with T.

Algorithm 2 :MCU-ELM

```
1: $w^{mc} = \begin{bmatrix} w^{mc}_{0,1} & \dots & w^{mc}_{0,h} \\ \vdots & \dots & \vdots \\ w^{mc}_{p_h-1,1} & \dots & w^{c}_{p_h-1,h} \end{bmatrix}_{n \times h}$
2: $H = \begin{bmatrix} P_{1,0}\left(g(a^{mc}_{1,1}), \dots, g(a^{mc}_{1,h})\right) \dots P_{1,p_o-1}\left(g(a^{mc}_{1,1}), \dots, g(a^{mc}_{1,h})\right) \\ \vdots \dots \vdots \\ P_{N,0}\left(g(a^{mc}_{N,1}), \dots, g(a^{mc}_{N,h})\right) \dots P_{N,p_o-1}\left(g(a^{mc}_{N,1}), \dots, g(a^{mc}_{N,h})\right) \end{bmatrix}_{N \times p_o}$
3: $T = \begin{bmatrix} t_{1,1} & \dots & t_{1,m} \\ \vdots & \dots & \vdots \\ t_{N,1} & \dots & t_{N,m} \end{bmatrix}_{N \times m}$
4: $\beta^{mc} = H^{\dagger}T$
```

3.5. The GA

The GA originally developed by Holland (1975) and his colleagues was inspired by Charles Darwin's biological evolution theory. It is a stochastic approach that can be used in many optimization problems. A possible solution to an optimization problem is encoded as a chromosome, and each parameter is encoded as a gene. The population's individuals are evaluated using a fitness function. The best ones have the highest probability of being selected for reproduction, producing the offspring of the evolution process. Afterward, a small percentage of the offspring undergoes a mutation procedure involving random chromosome changes. The process repeats until specific ending criteria are met. The structure of a typical GA can be seen in Algorithm 3 (Mirjalili, 2019; Kramer & Kramer, 2017).

It begins by creating the initial population, which contains a series of possible solutions (line 1). The original GA by Holland encoded each solution to a binary string (binary encoding), with each bit representing one gene. Alternative encoding schemes include permutation, real value, and tree encoding. In permutation encoding, the chromosome contains integer values representing positions in a sequence, and real value encoding has real values. Real value encoding is a popular scheme adopted in GAs for optimizing neural network weights. An individual in tree encoding represents a tree of functions or commands and is a popular choice in evolving programs or expressions (Katoch et al., 2021).

Algorithm 3 : GA

```
1: $Population \leftarrow create(Pop_{no})$
2: loop
3:     $Population_{evaluate} = evaluate(Population)$
4:     if $(criterion_{stop} = true)$ or $(solution_{best} = true)$ then
5:         $solution_{best} \leftarrow best(Population_{evaluate})$
6:         return $solution_{best}$
7:     end if
8:     $Population_{select} \leftarrow select(Population_{evaluate})$
9:     $Population_{crossover} \leftarrow crossover(Population_{select})$
10:    $Population \leftarrow mutation(Population_{crossover})$
11: end loop
```

After creating the initial population, the evolution process begins (line 2), where the initial population is evaluated according to a fitness function (line 3). The following line checks if the stopping criteria have been achieved or whether the optimal solution has been found. In case this condition is satisfied, the selected optimal solution (line 5) is returned (line 6). If the stopping criteria are unsatisfied, the evolution process continues with the selection operator (line 8) (Konak et al., 2006).

Natural selection allows fitter individuals to have higher probabilities for survival and reproduction. The roulette wheel selection mechanism simulates this idea by assigning selection probabilities to each chromosome according to their fitness values. Fitter chromosomes have higher chances for reproduction, while chromosomes with low fitness scores have low chances of getting selected. Low-fitness chromosomes are not excluded in order to increase the population's diversity. Other selection mechanisms include tournament, rank, and Boltzmann selection (Mirjalili, 2019).

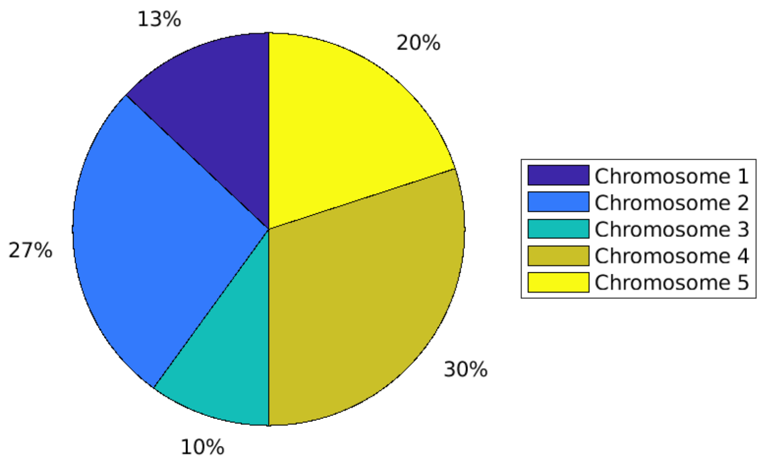

Figure 4: Roulette wheel selection mechanism. In the roulette wheel selection operator, all chromosomes are assigned a selection percentage proportional to their fitness value. The above pie chart visualizing the selection probabilities of five chromosomes shows that the fourth chromosome has the highest selection probability (30%).

The crossover procedure (line 9) reproduces the parent chromosomes selected in the previous phase. The one-point crossover operator depicted in Fig. 5 chooses a random crossover point and mutually exchanges genetic information between the two parent chromosomes. This way, two offspring are produced. Alternative reproduction processes involve multi-point, uniform, and masked crossover operators (Mirjalili, 2019).

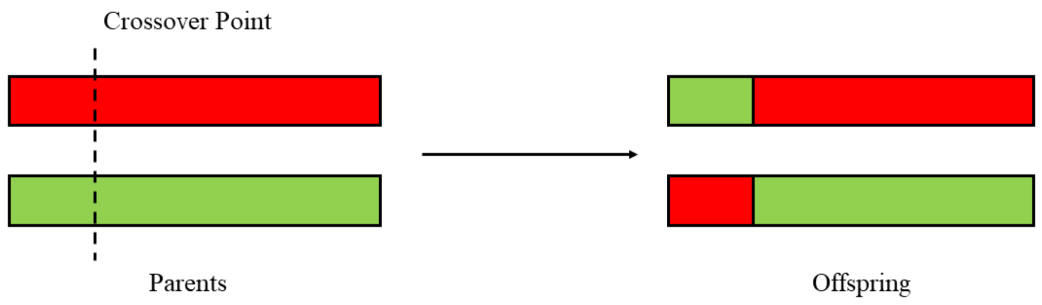

Figure 5: One-point crossover. The one-point crossover operation involves a mutual exchange of information between two parent chromosomes. The result of this procedure is the creation of two offspring.

Finally, the role of the mutation operator (line 10) is to keep the population diverse and help the GA avoid local solutions by altering one or more genes from a small number of randomly selected offspring. The bit mutation operator utilized in binary encoding where each mutated bit is changed from 0 value to 1 and vice-versa is depicted in Fig. 6 (Mirjalili, 2019).

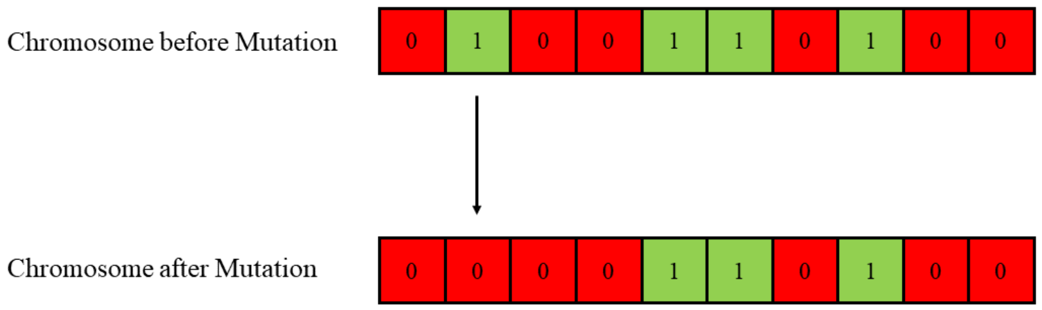

Figure 6: Bit mutation. The bit mutation operation changes one or more randomly selected genes in a binary string-encoded offspring chromosome. The value changes from 1 to 0 and vice-versa.

3.6. The EHO-ELM algorithm

The EHO-ELM algorithm hybridizes MCU-ELM with a modified GA. This method aims to create an optimal SLNN with MCUs in all its layers by tuning its hidden layer weights and finding an optimal sub-cube combination for all its neurons. The optimized MCU-ELM network will solve a specific classification or regression problem. The structure of EHO-ELM is described in Algorithm 4.

Algorithm 4 : EHO-ELM

```
 1: $Population \leftarrow create(Pop_{no})$
 2: loop
 3:     $Population_{evaluate} = evaluate(Population)$
 4:     $solution_{best} \leftarrow best(Population_{evaluate})$
 5:     if $solution_{best}$ is unchanged for 5 generations then
 6:         return $solution_{best}$
 7:     end if
 8:     $Population_{select} \leftarrow select(\frac{Population_{evaluate}}{2})$
 9:     $Population_{crossover} \leftarrow crossover(Population_{select})$
10:     $Population \leftarrow mutation(Population_{crossover})$
11: end loop
```

EHO-ELM begins by creating the initial population containing a set of SLNNs with MCUs in all their layers (line 1). Each chromosome in the initial population represents an SLNN. It encodes two types of information: the weight values for the hidden layer MCUs and the MCU structure from both its layers (hidden and output). The information regarding each MCU is encoded as a gene in the chromosome with a 2^5 maximum sub-cube size for every MCU. The population's size (Pop_{no}) is calculated automatically using equation 6 by taking into consideration the number of hidden neurons (h).

$$Pop_{no} = \begin{cases} 50, & 2h > 50 \\ 2h, & 0 > 2h \leq 50 \end{cases} \tag{6}$$

After creating the initial population, the evolution process starts (line 2) with the first generation of the modified GA. The evaluation process (line 3) assigns a fitness value to each chromosome. This value is computed by training all SLNNs using MCU-ELM and calculating their classification accuracy (acc) in classification problems or mean square error (MSE) in regression problems. The SLNNs having the highest acc or the lowest MSE values are considered fitter. The acc values are calculated using formula 7 where k denotes the folds' number. The symbol pat defines the total number of input patterns, while err defines the erroneously classified patterns regarding the current fold.

$$acc = \frac{1}{k}\sum_{i=1}^{k}\left(1 - \frac{err}{pat}\right). \tag{7}$$

The MSE values are calculated using formula 8 where k denotes the folds' number and pat is the input patterns' number. The j target network output value and the j network output value for fold i are defined using terms t_i^j and y_i^j.

$$MSE = \frac{1}{kpat}\sum_{i=1}^{k}\Big(\sum_{j=1}^{pat}(t_i^j - y_i^j)^2\Big) \tag{8}$$

In line 4, the best solution is selected according to the fitness values calculated in the previous step. If the best solution remains the same for five iterations of the GA (generations), as seen in line 5, it returns the best network found (line 6). Alternatively, the evolution process continues with selecting 50% of the population (line 8) for reproduction. The selection is made according to their fitness values calculated in step 3.

The reproduction process uses a custom crossover operator that creates the offspring by mutually exchanging sub-cubes between two parent chromosomes (Fig 7).

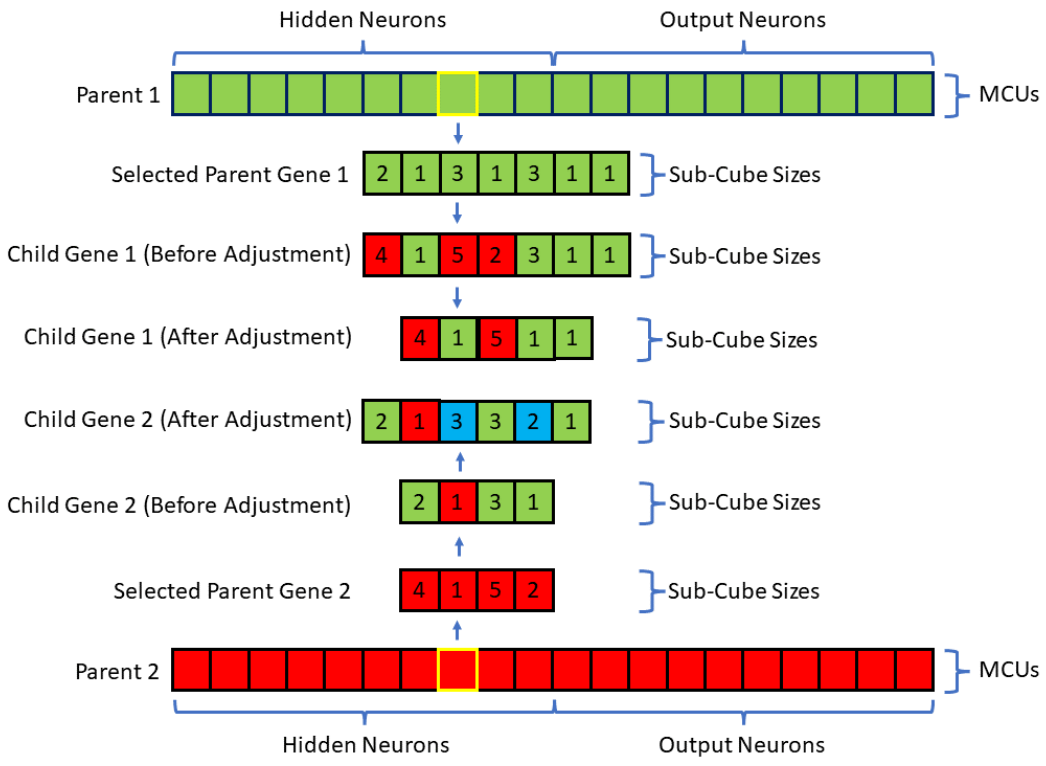

Figure 7: EHO-ELM crossover operator. The EHO-ELM crossover operator mutually exchanges information between two parent chromosomes depicted with red and green colors. The exchange uses equal probability between the two parent individuals and may produce offspring with different MCU input sizes. Due to this reason, an adjustment mechanism is utilized. The adjustment mechanism automatically creates (blue color) or removes genes for retaining offspring validity. In this example, the maximum acceptable sub-cube size is 5.

The exchange is done according to a probability that is the same for both parent sub-cubes. This procedure might create MCUs with different input sizes than the parent ones. Due to this reason, the custom crossover operator automatically adjusts the offspring chromosomes. Suppose an MCU in the offspring chromosome contains more inputs. In that case, the operator automatically removes random sub-cubes from the MCU until the number of inputs becomes the same or less than the parent one. Suppose the number of inputs is less than the parent MCU and the difference between the parent and child one is higher than the maximum allowed sub-cube size. In that case, the operator creates a random sub-cube containing a randomly created input size and weights. The created MCU is then placed in a random location inside the child MCU. This procedure is repeated until the difference in the number of inputs between the parent and child genes becomes equal to or less than the maximum allowed sub-cube input size. If the last condition is satisfied, a random sub-cube is created with the inputs taken from the difference between the parent and child genes. This sub-cube is then placed in a random location inside the child gene. The above procedure creates valid SLNNs and tunes the hidden layer weights.

A few hidden layer neurons are randomly selected for mutation. The self-adaptive mutation operator changes the weight values from one sub-cube for each selected MCU. It is defined in equation 9 and works by considering the fitness value of the best chromosome found at each generation. Suppose it is the first generation ($generation = 1$) of the evolution process or a better chromosome has been found ($(generation > 1) \wedge (fitness_{current} > fitness_{previous})$). In that case, the mutation rate receives its lowest value (10%). If the crossover operation has not produced better offspring, then the mutation rate increases by 20% ($\mu_{rate} = \mu_{rate} + 20\%$) until it reaches its maximum value (50%).

$$\mu_{rate} = \begin{cases} \mu_{rate} = 10\%, & (generation = 1) \vee ((generation > 1) \\ & \wedge (fitness_{current} > fitness_{previous})) \\ \mu_{rate} = \mu_{rate} + 20\%, & fitness_{current} \leq fitness_{previous} \\ \mu_{rate} = 50\%, & \mu_{rate} \geq 50\% \end{cases} \tag{9}$$

4. EHO-ELM for PD detection

The proposed EHO-ELM-based method visualized in Fig. 8 utilizes the "Parkinson's disease classification" dataset (Sakar et al., 2018) from UCI machine learning repository (Dua & Graff, 2017) containing 753 extracted features from the voice recordings of 188 patients (107 male and 81 female) having an age range between 33 and 87 years. The control group contained 64 healthy individuals (23 male and 41 female) aged between 41 and 82.

The dataset contained one additional column containing each participant's ID number, which has been removed.

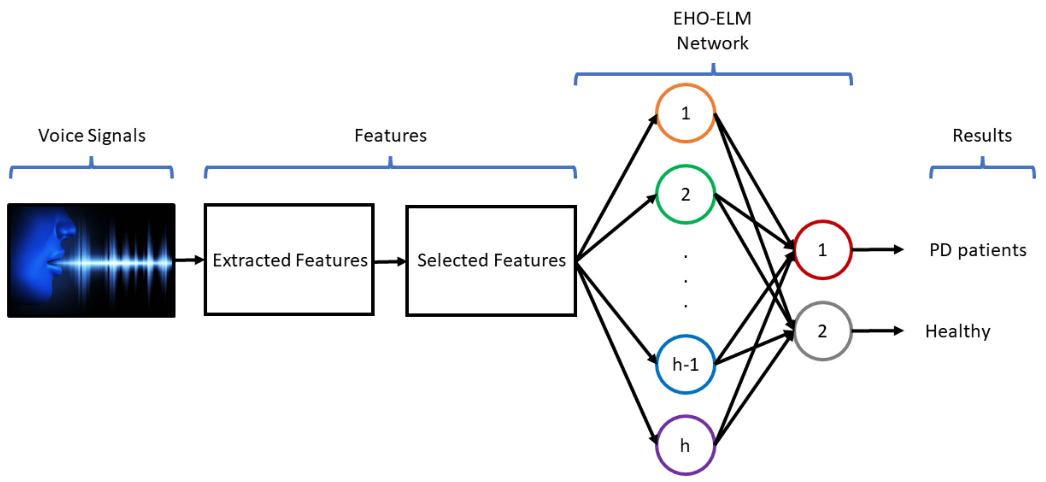

Figure 8: EHO-ELM for PD detection system architecture. The voice signals are recorded from 188 non-healthy and 64 healthy individuals. A series of feature extraction processes categorized into six sets creates 753 features. The MRMR features selection process is utilized to select the 50 best ones, which form the input vector to an EHO-ELM trained SLNN with h hidden neurons responsible for classifying the participants into two categories (PD patients and normal).

The "Parkinson's disease classification" dataset contains six extracted feature sets. The first set includes a group of features termed "baseline features" (jitter, shimmer, fundamental frequency parameters, harmonicity parameters, recurrence period density entropy, detrended fluctuation analysis, and pitch period entropy), popular choices for PD feature extraction. The second group, termed "time-frequency features" (speech intensity, formant frequencies, and bandwidth-based), contains features from the speech signals' spectrograms. The third group has extracted features based on the Mel-frequency cepstral coefficients that closely resemble the human ear's efficient filtering abilities. The fourth group contains wavelet transform-based features received from speech samples' raw fundamental frequency contour. The fifth set has vocal fold features, while the final feature set is extracted using Q-factor wavelet transform. The latter is a fully discrete and overcomplete wavelet transform method Sakar et al. (2018).

The dataset underwent a feature selection method using the MRMR algorithm, which selected the 50 best features. MRMR selects a sub-feature set containing the maximum correlation with the class (output) and the minimum correlation between the selected features (Ding & Peng, 2005). The selected feature subset is then introduced as input to the EHO-ELM algorithm, classifying the subjects into two categories (non-healthy and healthy).

5. Experimental results

The EHO-ELM algorithm was tested with ten BP variants (BFGS, PBCG, FRCG, PRCG, SCG, SD, GDM, GDMALR, OSS, and RPROP), SVM, DT, ELM, and two ELM-based approaches (OS-ELM and MCU-ELM). The "Parkinson's disease classification" dataset was initially divided with 80% of the data forming the training set, and the rest 20% was reserved as a separate test set. This setup was utilized in the SVM, DT, ELM, and all ELM-based variants. The training set was further divided into training and validation sets using 10-fold cross-validation. All BP-based approaches used the training/validation/test setup to find the optimal number of training epochs and avoid over-fitting. It was also used in the proposed EHO-ELM algorithm for calculating each individual's fitness using unknown (validation) data.

Table 1: Experimental Setup Parameters

Parameter Name	**Symbol**	**Values/Types**
Linear Neuron Weights	w^l	$[-1,1]^n, n \in \mathbb{N}^*$
Multi-Cube Neuron Weights	w^{mc}	$[-1,1]^{p_{no}}, p_{no} \in \mathbb{N}^*$
Inputs	x	$[-1,1]^n, n \in \mathbb{N}^*$
Hidden Layer Neurons' Transfer Function	g	*sigmoid*
Output Layer Neurons' Transfer Function	g	*identity*
Hidden Layer Neurons No	h	50
Folds No	k	10
Experiment Sets	$expNo$	10
MCU-ELM's MCUs Sub-Cube Dimension	$d_j^{MCU-ELM}$	$1, j \in \mathbb{N}^*$
EHO-ELM's MCUs Sub-Cube Dimension	$d_j^{EHO-ELM}$	$[1,2,\ldots,5], j \in \mathbb{N}^*$

The parameters utilized for the experiments' execution are summarized in Table 1. In the neural network-based methods, all weights and thresholds for all neuron types (traditional linear neurons and MCUs) were randomized inside the $[-1,1]$ interval using real values from the uniform distribution.

The number of hidden layer units was fixed to 50 nodes, with all of them having the *sigmoid* transfer function. The activation function for the output layer units was the *identity* function. The experiments in all neural network-based methods were repeated ten times to avoid bias in the results due to the random initialization of the hidden layer weights and thresholds. The MCU-ELM-trained network contained one-dimensional sub-cubes (2^1) in all MCUs, while the EHO-ELM-trained network's sub-cube dimension varied from 1 to 5. Finally, the radial basis function (RBF) kernel was selected for the SVM algorithm.

The comparison of EHO-ELM with the 15 methods mentioned above in terms of classification accuracy is visualized in Table 2. It can be seen that the proposed EHO-ELM-based method got the highest *acc* (87.55%) compared to the other existing machine learning methods.

Table 2: Comparison Results

BFGS	**PBCG**	**FRCG**	**PRCG**	**SCG**	**GD**	**GDM**	**GDMALR**
69.45%	66.45%	69.91%	66.86%	65.43%	57.87%	57.58%	64.91%

OSS	**RPROP**	**SVM**	**DT**	**OS-ELM**	**ELM**	**MCU-ELM**	**EHO-ELM**
68.53%	86.78%	86.84%	68.42%	86.45%	86.71%	86.45%	87.55%

The Wilcoxon signed-ranked test was adopted for evaluating the statistical significance of the results presented in Table 2. It is a non-parametric statistical hypothesis test that uses two matched samples to compare the locations of two populations or assess a population's location based on a data sample (Wilcoxon; Conover, 1999). The result from this test is a probability (p-value) that the two compared populations are identical. According to Wilcoxon signed-rank test results presented in Fig. 9 from comparing EHO-ELM with the 15 machine learning methods, EHO-ELM's results are statistically significant since the p-value from all compared methods was less than 5%.

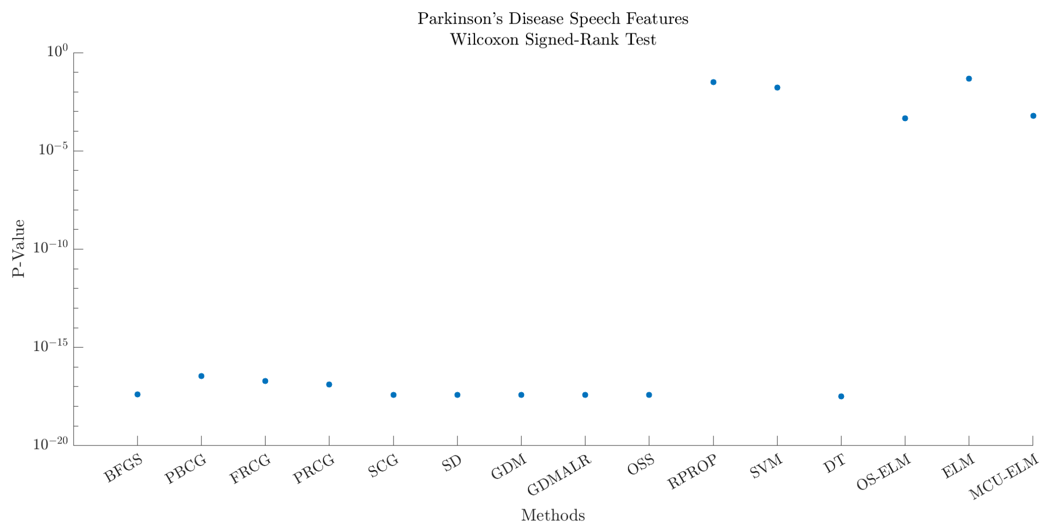

Figure 9: Wilcoxon signed-rank test results. The results from the Wilcoxon signed-rank test between the proposed EHO-ELM-based method and 15 machine learning methods in the "Parkinson's disease classification" dataset are depicted in this logarithmic plot. All the p-values in the compared methods were less than 5%, indicating that the results were statistically significant.

6. Discussion

An automated PD diagnosis tool from speech signals can be a valuable asset for health specialists because it can help them in a more accurate early diagnosis of the disease since only 85% of the cases are diagnosed correctly by PD specialists (Rizzo et al., 2016). PD patients have a long lifespan, and besides the pathological problems (tremors, stiffness in the patient's limbs, gait and balance issues), PD patients also face psychological problems like anxiety and depression (Marie, 2020; Factor & Weiner, 2007). Due to these severe side effects, early diagnosis of the disease is important because it can increase patients' quality of life using anti-parkinsonian medication. Studies have shown that patients who received anti-parkinsonian medicines have a better quality of life than untreated ones (Grosset et al., 2007; Rees et al., 2018).

The EHO-ELM hybrid algorithm was selected as a classifier for the "Parkinson's disease classification" dataset. It has an evolution process that automatically optimizes the hidden layer weights and finds an optimal sub-cube combination for all network neurons. At the same time, all parameters of the evolution process are self-adaptive. One reason for selecting EHO-ELM is that it retains ELM's simplicity and requires minimum parameter tuning from the user's perspective (the user needs only to choose the number of hidden layer units). The experimental results in the previous section showed that EHO-ELM achieved the highest accuracy percentage compared to 15 machine learning methods. The significance of these results was verified using the Wilcoxon sign-rank statistical test, where the p-value was less than 5% in all cases.

EHO-ELM is a hybrid algorithm combining MCU-ELM with a modified GA for improving MCU-ELM's generalization performance. Although this is evident in the experimental results in Table 2, it comes with a significant performance cost. The evolution process requires many SLNNs to be trained with MCU-ELM at every iteration (generation) of the GA. The training process is computationally intensive and time-consuming. The solution to this problem was adopting a parallel processing strategy for the EHO-ELM algorithm, enabling it to work in multi-core systems by using a parallel GA that can significantly reduce training time.

7. Conclusion

The proposed hybrid MCU-ELM-based approach was selected as a classifier for the "Parkinson's disease classification" dataset containing entries from speech signals. These signals were received from PD patients and normal ones. The signals underwent a feature extraction and selection stage with 50 features forming the input vector to the EHO-ELM algorithm. This algorithm was selected for the classification task between PD and normal individuals because it can optimize the hidden layer weights and find an optimal combination of sub-cube units for all network neurons. The higher-order units by Gurney (1989) were chosen due to their advanced generalization performance in ELM-based trained SLNNs as was experimentally shown in earlier works from Christou (2023); Christou et al. (2023).

Section 5 compared EHO-ELM with 15 existing methods and experimentally verified that the proposed method is better regarding *acc*. These results were statistically significant according to the Wilcoxon signed-ranked test. Finally, future work involves converting HO-ELM to multi-layer networks and creating a larger PD dataset containing a higher number of participants' speech signals.

CRediT authorship contribution statement

Vasileios Christou: Conceptualization, Methodology, Software, Writing – original draft, Investigation, Resources. **Alexandros T. Tzallas:** Visualization, Validation, Writing – review & editing, Supervision, Project administration. **Georgios Tsoumanis:** Validation, Supervision, Investigation. **Markos G. Tsipouras:** Data curation, Validation, Writing – review & editing, Supervision, Project administration. **Dimitrios Dimopoulos:** Validation, Supervision, Investigation. **Dimitrios Varvarousis:** Validation, Supervision, Investigation. **Avraam Ploumis:** Validation, Writing – review & editing, Supervision, Project administration. **Nikolaos Giannakeas:** Formal analysis, Validation, Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

We acknowledge support of this work by the project "MEGATRON" (MIS 5047227) which is implemented under the Action "Reinforcement of the Research and Innovation Infrastructure", funded by the Operational Programme "Competitiveness, Entrepreneurship and Innovation" (NSRF 2014-2020) and co-financed by Greece and the European Union (European Regional Development Fund).

References

Agarwal, A., Chandrayan, S., & Sahu, S. S. (2016). Prediction of parkinson's disease using speech signal with extreme learning machine. In *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)* (pp. 3776–3779). IEEE.

Alhussein, M., & Muhammad, G. (2018). Voice pathology detection using deep learning on mobile healthcare framework. *IEEE Access*, *6*, 41034–41041.

Anter, A. M., Mohamed, A. W., Zhang, M., & Zhang, Z. (2023). A robust intelligence regression model for monitoring parkinson's disease based on speech signals. *Future Generation Computer Systems*, *147*, 316–327.

Asmae, O., Abdelhadi, R., Bouchaib, C., Sara, S., & Tajeddine, K. (2020). Parkinson's disease identification using knn and ann algorithms based on voice disorder. In *2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)* (pp. 1–6). IEEE.

Battiti, R. (1992). First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, *4*, 141–166.

Broyden, C. G. (1970a). The convergence of a class of double-rank minimization algorithms: 1. general considerations. *IMA Journal of Applied Mathematics*, *6*, 76–90.

Broyden, C. G. (1970b). The convergence of a class of double-rank minimization algorithms: 2. the new algorithm. *IMA journal of applied mathematics*, *6*, 222–231.

Cauchy, A. et al. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, *25*, 536–538.

Chen, H.-L., Wang, G., Ma, C., Cai, Z.-N., Liu, W.-B., & Wang, S.-J. (2016). An efficient hybrid kernel extreme learning machine approach for early diagnosis of parkinson's disease. *Neurocomputing*, *184*, 131–144.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T. et al. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*, 1–4.

Christou, V. (2023). Higher-order extreme learning machine. Unpublished.

Christou, V., Tzallas, A. T., Tsipouras, M. G., Giannakeas, N., & Tsoumanis, G. (2023). Evolutionary higher-order extreme learning machine. Unpublished.

Conover, W. J. (1999). *Practical nonparametric statistics* volume 350. john wiley & sons.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273–297.

Das, P., & Nanda, S. (2023). Bio-inspired voting ensemble weighted extreme learning machine classifier for the detection of parkinson's disease. *Research on Biomedical Engineering*, (pp. 1–15).

Despotovic, V., Skovranek, T., & Schommer, C. (2020). Speech based estimation of parkinson's disease using gaussian processes and automatic relevance determination. *Neurocomputing*, *401*, 173–181.

Ding, C., & Peng, H. (2005). Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, *3*, 185–205.

Dua, D., & Graff, C. (2017). UCI machine learning repository. URL: `http://archive.ics.uci.edu/ml`.

Erdogdu Sakar, B., Serbes, G., & Sakar, C. O. (2017). Analyzing the effectiveness of vocal features in early telediagnosis of parkinson's disease. *PloS one*, *12*, e0182428.

Factor, S. A., & Weiner, W. J. (2007). Parkinson's disease: diagnosis and clinical management, .

Fletcher, R. (1970). A new approach to variable metric algorithms. *The computer journal*, *13*, 317–322.

Fletcher, R., & Reeves, C. M. (1964). Function minimization by conjugate gradients. *The computer journal*, *7*, 149–154.

García-Ordás, M. T., Benítez-Andrades, J. A., Aveleira-Mata, J., Alija-Pérez, J.-M., & Benavides, C. (2023). Determining the severity of parkinson's disease in patients using a multi task neural network. *Multimedia Tools and Applications*, (pp. 1–16).

Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of computation*, *24*, 23–26.

Grosset, D., Taurah, L., Burn, D., MacMahon, D., Forbes, A., Turner, K., Bowron, A., Walker, R., Findley, L., Foster, O. et al. (2007). A multicentre longitudinal observational study of changes in self reported health status in people with parkinson's disease left untreated at diagnosis. *Journal of Neurology, Neurosurgery & Psychiatry*, *78*, 465–469.

Guatelli, R., Aubin, V., Mora, M., Naranjo-Torres, J., & Mora-Olivari, A. (2023). Detection of parkinson's disease based on spectrograms of voice recordings and extreme learning machine random weight neural networks. *Engineering Applications of Artificial Intelligence*, *125*, 106700.

Gurney, K. N. (1989). *Learning in networks of structured hypercubes*. Ph.D. thesis Brunel University London, UK.

Gürüler, H. (2017). A novel diagnosis system for parkinson's disease using complex-valued artificial neural network with k-means clustering feature weighting method. *Neural Computing and Applications*, *28*, 1657–1666.

Hireš, M., Gazda, M., Drotar, P., Pah, N. D., Motin, M. A., & Kumar, D. K. (2022). Convolutional neural network ensemble for parkinson's disease detection from voice recordings. *Computers in biology and medicine*, *141*, 105021.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Huang, G.-B., Liang, N.-Y., Rong, H.-J., Saratchandran, P., & Sundararajan, N. (2005). On-line sequential extreme learning machine. *Computational Intelligence*, *2005*, 232–237.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)* (pp. 985–990). Ieee volume 2.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, *70*, 489–501.

Jatoth, C., Neelima, E., Mayuri, A., & Annaluri, S. R. (2022). Effective monitoring and prediction of parkinson disease in smart cities using intelligent health care system. *Microprocessors and Microsystems*, *92*, 104547.

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, *80*, 8091–8126.

Konak, A., Coit, D. W., & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, *91*, 992–1007.

Kramer, O., & Kramer, O. (2017). *Genetic algorithms*. Springer.

Little, M., McSharry, P., Hunter, E., Spielman, J., & Ramig, L. (2008). Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *Nature Precedings*, (pp. 1–1).

Marie, L. (2020). *The Complete Guide for People With Parkinson's Disease and Their Loved Ones*. Purdue University Press.

Meghraoui, D., Boudraa, B., Merazi-Meksen, T., & Vilda, P. G. (2021). A novel pre-processing technique in pathologic voice detection: Application to parkinson's disease phonation. *Biomedical Signal Processing and Control*, *68*, 102604.

Meza, J. C. (2010). Steepest descent. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*, 719–722.

Mirjalili, S. (2019). Evolutionary algorithms and neural networks. In *Studies in computational intelligence*. Springer volume 780.

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, *6*, 525–533.

Narendra, N., Schuller, B., & Alku, P. (2021). The detection of parkinson's disease from speech using voice source information. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *29*, 1925–1936.

Orozco-Arroyave, J. R., Arias-Londoño, J. D., Vargas-Bonilla, J. F., Gonzalez-Rátiva, M. C., & Nöth, E. (2014). New spanish speech corpus database for the analysis of people suffering from parkinson's disease. In *LREC* (pp. 342–347).

Parkinson, J. (2002). An essay on the shaking palsy. *The Journal of Neuropsychiatry and Clinical Neurosciences*, *14*, 223–236. doi:10.1176/jnp.14.2.223. PMID: 11983801.

Peker, M. (2016). A decision support system to improve medical diagnosis using a combination of k-medoids clustering based attribute weighting and svm. *Journal of medical systems*, *40*, 116.

Polak, E., & Ribiere, G. (1969). Note sur la convergence de méthodes de directions conjuguées. *Revue française d'informatique et de recherche opérationnelle. Série rouge*, *3*, 35–43.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, *4*, 1–17.

Powell, M. J. D. (1977). Restart procedures for the conjugate gradient method. *Mathematical programming*, *12*, 241–254.

Rees, R. N., Acharya, A. P., Schrag, A., & Noyce, A. J. (2018). An early diagnosis is not the same as a timely diagnosis of parkinson's disease. *F1000Research*, *7*.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE international conference on neural networks* (pp. 586–591). IEEE.

Rizzo, G., Copetti, M., Arcuti, S., Martino, D., Fontana, A., & Logroscino, G. (2016). Accuracy of clinical diagnosis of parkinson disease: a systematic review and meta-analysis. *Neurology*, *86*, 566–576.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*, 211–252.

Sakar, B. E., Isenkul, M. E., Sakar, C. O., Sertbas, A., Gurgen, F., Delil, S., Apaydin, H., & Kursun, O. (2013). Collection and analysis of a parkinson speech dataset with multiple types of sound recordings. *IEEE Journal of Biomedical and Health Informatics*, *17*, 828–834.

Sakar, C., Serbes, G., Gunduz, A., Nizam, H., & Sakar, B. (2018). Parkinson's Disease Classification. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5MS4X.

Sakar, C. O., Serbes, G., Gunduz, A., Tunc, H. C., Nizam, H., Sakar, B. E., Tutuncu, M., Aydin, T., Isenkul, M. E., & Apaydin, H. (2019). A comparative analysis of speech signal processing algorithms for parkinson's disease classification and the use of the tunable q-factor wavelet transform. *Applied Soft Computing*, *74*, 255–263.

Samui, P., Roy, S. S., & Balas, V. E. (2017). *Handbook of neural computation*. Academic Press.

Scales, L. (1985). *Introduction to non-linear optimization*. Springer-Verlag.

Schweitzer, R. C., Morris, J. B., & MD, A. R. L. A. P. G. (2000). A tutorial on neural networks using the broyden-fletcher-goldfarb-shanno (bfgs) training algorithm and molecular descriptors with application to the prediction of dielectric constants through the development of quantitative structure property relationships (qsprs). *United States Army Research Laboratory ARL-TR-2155*, .

Shahid, A. H., & Singh, M. P. (2020). A deep learning approach for prediction of parkinson's disease progression. *Biomedical Engineering Letters*, *10*, 227–239.

Shahsavari, M. K., Rashidi, H., & Bakhsh, H. R. (2016). Efficient classification of parkinson's disease using extreme learning machine and hybrid particle swarm optimization. In *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)* (pp. 148–154). IEEE.

Shanno, D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, *24*, 647–656.

Tunc, H. C., Sakar, C. O., Apaydin, H., Serbes, G., Gunduz, A., Tutuncu, M., & Gurgen, F. (2020). Estimation of parkinson's disease severity using speech features and extreme gradient boosting. *Medical & Biological Engineering & Computing*, *58*, 2757–2773.

Van Den Eeden, S. K., Tanner, C. M., Bernstein, A. L., Fross, R. D., Leimpeter, A., Bloch, D. A., & Nelson, L. M. (2003). Incidence of parkinson's disease: variation by age, gender, and race/ethnicity. *American journal of epidemiology*, *157*, 1015–1022.

Vapnik, V. (1998). Statistical learning theory.

Von Winterfeldt, D., & Edwards, W. (1986). Decision analysis and behavioral research. *(No Title)*, .

Wang, Y., Wang, A.-N., Ai, Q., & Sun, H.-J. (2017). An adaptive kernel-based weighted extreme learning machine approach for effective detection of parkinson's disease. *Biomedical Signal Processing and Control*, *38*, 400–410.

Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA*, .

Wilcoxon, F. (). Individual comparisons by ranking methods, biometrics bulletin 1 (1945) 80–83. *URL: http://www.jstor.org/stable/3001968.doi*, *10*, 3001968.

Wodzinski, M., Skalski, A., Hemmerling, D., Orozco-Arroyave, J. R., & Nöth, E. (2019). Deep learning approach to parkinson's disease detection using voice recordings and convolutional neural network dedicated to image classification. In *2019 41st annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (pp. 717–720). IEEE.

Xue, Z., Lu, H., Zhang, T., Xu, J., & Guo, X. (2023). A local dynamic feature selection fusion method for voice diagnosis of parkinson's disease. *Computer Speech & Language*, (p. 101536).