IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 December 2023, accepted 11 February 2024, date of publication 19 February 2024, date of current version 23 February 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3367588*

RESEARCH ARTICLE

# Neurodegenerative Condition Detection Using Modified Metaheuristic for Attention Based Recurrent Neural Networks and Extreme Gradient Boosting Tuning

**JELICA CINCOVIC[1], LUKA JOVANOVIC[2], BOSKO NIKOLIC[1], AND NEBOJSA BACANIN[3,4]**
[1]School of Electrical Engineering, University of Belgrade, 11120 Belgrade, Serbia
[2]Faculty of Technical Sciences, Singidunum University, 11000 Belgrade, Serbia
[3]Department of Informatics and Computing, Singidunum University, 11000 Belgrade, Serbia
[4]EU Research Unit, Middle East University, Amman 11831, Jordan

Corresponding author: Nebojsa Bacanin (nbacanin@singidunum.ac.rs)

This work was supported by the Science Fund of the Republic of Serbia under Grant 6526093.

**ABSTRACT** Parkinson's disease is a neurological disorder, caused by the death of dopaminergic neurons which can cause various movement disorders to appear, recognized as standard Parkinson's motor symptoms. A drug to stop the progression of the disease is very difficult to find, so current treatment is based on alleviating the symptoms of the disease itself. As no direct treatment exists that would cure the condition, early detection and proper treatment are essential in maintaining the patient's quality of life. This work explores the potential of merging artificial intelligence and machine learning algorithms for Parkinson's disease early detection from finger-tapping accelerometer tests. Time series classification is explored through the use of recurrent neural networks augmented with and without attention layers. Additionally, extreme gradient boosting in combination with statistical analysis is explored in order to differentiate Parkinson's from other developing neurodegenerative disorders. As the performance of algorithms hinges on proper parameter selection, this work applies metaheuristics for performance optimization. A modified version of a recently introduced sinh cosh optimizer algorithm is also proposed. The approach is tested on a publicly available real-world clinical dataset consisting of patients and control group samples and a total of three separate experiments were conducted. The introduced optimizer demonstrated admirable performance in comparative analysis, with the best performing models exceeding 90% accuracy.

**INDEX TERMS** Parkinson's disease, metaheuristics, sinh cosh optimization, optimization, hyperparamater tuning.

## I. INTRODUCTION

Parkinson's disease (PD) is a neurological disorder, caused by the death of dopaminergic neurons. As a consequence of a lack of dopamine, various movement disorders appear, recognized as standard Parkinsonian motor symptoms. These symptoms include rest tremors, bradykinesia, muscular rigidity, and postural and gait impairment [1]. In addition to the already classic symptoms, some patients also experience other symptoms such as early dementia, ataxia, or frequent falls that may indicate the existence of other atypical Parkinsonian disorders. Most common atypical parkinsonian disorders are multiple system atrophy (MSA), progressive supranuclear palsy (PSP), corticobasal degeneration (CBD) as well as other rarer causes [2].

A drug to stop the progression of the disease is very difficult to find, so current treatment is based on alleviating the symptoms of the disease itself with drugs that raise dopamine levels. However, it has been shown that lack of dopamine is not the only thing that needs to be solved, but that other neurotransmitters are also involved in PD. Research has also shown that genetics, which has a significant

The associate editor coordinating the review of this manuscript and approving it for publication was Valentina E. Balas.

influence, must be added to the environmental factors that have long been considered the main causes of the disease. Furthermore, attention must be paid to atypical Parkinsonian disorders, which can often be misdiagnosed as PD. Treatment of these diseases with standard methods used to treat PD can further worsen the condition of patients. All of the above significantly affects the complexity of the disease, which begins to develop years before the first symptoms, and early diagnosis is of key importance for the most successful treatment of patients.

There is no definitive test that would confirm with certainty the existence of PD or other atypical Parkinsonian disorders. This is exactly the reason why scientists make their contribution in this field, trying to correctly predict the existence of mentioned diseases. Some of the ways of detecting the disease are finger-tapping or gait analysis, Magnetic Resonance Imaging (MRI) scans, electroencephalography (EEG) tests, etc. Such comprehensive analyses can be improved with machine learning algorithms, which would contribute to both accuracy and saving resources.

Machine learning makes it possible to see very complex dependencies between data, which might be missed by the human eye with standard data analysis methods. The main categories of machine learning are divided based on the data which are given to the algorithms, and they are: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. In supervised learning, which is used in this paper, the data is labeled [3]. The main problems solved in supervised learning are classification and regression problems. There are many algorithms that deal with solving these problems [4], and each algorithm has hyperparameters that need to be chosen well because they significantly affect the performance of the algorithm itself. As the number of hyperparameters increases, finding the best values becomes more and more difficult. Outdated approaches such as trial and error have proven to be very inefficient and adequate solutions are needed for hyperparameter optimization, which is considered an NP-hard problem and cannot be solved using discreet methods [5].

Current state-of-the-art solutions for hyperparameter optimization are found in metaheuristic algorithms. A metaheuristic or a high-level procedure is used to produce the most optimal solution in the easiest and smartest way possible when standard mathematical methods fail [6]. Metaheuristics have found inspiration in nature, physics, and other elements in the environment. Some of the most popular nature-inspired optimizations are ant colony optimization, particle swarm optimization, cuckoo search, bacterial colony optimization, etc. Other metaheuristics based on physics phenomena are central force optimization, spiral optimization, gravitational search, etc. It has been shown that metaheuristics provide the most optimal solution in most cases, but they do not guarantee it. Even in cases where the most optimal solution is not found, metaheuristics will give a satisfactory answer, which is always better than being left without it in search of the best one.

The scientific contributions of this work can be summarized as the following:

- A proposal for time-series-based methods for detecting tremors associated with neurodegenerative disease with high precision
- A statistics-based approach combined with XGBoost to detect the specific kind of degenerative disease associated with the symptoms
- An introduction of a modified approach for early detection of Parkinson's disease in patients from finger-taping test data
- The proposal of a modified optimization metaheuristic used to select hyperparameters for detection algorithms and improve overall performance

The remainder of this work follows the here outlined structure: Section II presents related works that apply AI for tackling neurodegenerative disturbers. Section III presents the base algorithm followed by a detailed discussion of the introduced modification. In Section IV and Section V, the experimental setup is described in detail followed by the observed outcomes and their detailed discussion. Finally, a conclusion is provided alongside potential future works in Section VI

## II. RELATED WORKS

There are studies that have dealt with the detection of PD and atypical Parkinsonian disorders. Certain research has been devoted to the analysis of MRI scans. MRI has significantly contributed to the accuracy of diagnosing diseases. Analysis of 3T T1 weighted MRI scans [7] or neuromelanin sensitive magnetic resonance imaging (NMS-MRI) scans [8] with convolutional neural networks (CNN) proved to be one of the possible tools for early disease detection, with testing accuracies above 95% in case of weighted MRI scans and above 80% for NMS-MRI analysis. As access to MRI machines may not be easily accessible to everyone, other, less expensive approaches have been analyzed.

Gait analysis is also a factor that can significantly contribute to the early detection of the disease. The most well-known dataset with collected data on patients' gait has spawned several works. Gait analysis with the help of 1D CNN [9] gave an accuracy above 98% and using a Support Vector Machine (SVM) on the same data [10] achieved an accuracy of above 94%. Another way of detecting the existence of the disease is a finger-tapping test [11] that has yet to be fully explored for diagnosis.

Machine learning algorithms are used for increasingly successful analyses and predictions in the PD domain. For the most efficient solution, it is important to reduce features as much as possible so that the models are more optimal, fast, and precise. Some research in other fields has recognized this, so there are algorithms, that serve for feature reduction, based on Recurrent Neural Networks (RNN) [12] or long short-term memory networks (LSTM) [13] models, as is the case in this work.

In addition to reducing features, it is necessary to choose the appropriate model for solving classification problems, and the eXtreme Gradient Boosting (XGBoost) algorithm was selected. XGBoost is an algorithm that gives very high performance and predicts values for various problems with a high percentage of accuracy. Other works in the field of PD detection [14], [15] have also used this algorithm, and have obtained very satisfactory results, with accuracy percentages of over 85% and 90% respectively.

## A. ATTENTION BASED RECURRENT NEURAL NETWORKS

The human brain works incredibly, and scientists have tried to simulate the connections of neurons in the brain with artificial neural network (ANN) algorithms [16]. The architecture of an artificial neural network consists of an input layer of neurons, where the data is inserted, hidden layers, of which there may be more, depending on the problem being solved, and an output layer of neurons, in which our desired results are located. Neurons are interconnected by connections, which have a weight. The output values from the neuron are defined by an activation function that depends on the current value of the neuron, the weight, and the bias.

Recurrent neural networks are types of neural networks that have feedback and have proven to be excellent solutions to time series problems. This is due to the feedback mechanism allowing the network to remember and learn from previous inputs, unlike ordinary feedforward neural networks that do not have feedback connections. Several types of RNN networks tackled some of the associated issues observed in the original architecture such as exploding and vanishing gradients.

When the results from the model are obtained, it is possible to calculate the accuracy of the model and how much the error is, using the error function. The error function is the average of the value of the loss function on all data from the set, while the loss function is a measure of the deviation of the prediction from the correct value. The goal is to minimize the error (which can be done using a method called gradient descent), and that is why the backpropagation process is performed, which adjusts the network parameters (weights and biases) to reduce the error. In the context of working with RNN, backpropagation is done over the unfolded network and this process is called backpropagation through time (BPTT).

Precisely because of its property that the current output from the network is influenced by previous inputs, this kind of architecture is suitable for time series and sequences, and this is one of the main uses of RNN. RNNs can also be combined with other architectures and ML algorithms, to solve more complex problems and exploit the best aspects of all algorithms.

The attention phenomenon lacks a precise mathematical definition, and its incorporation into the Luong attention-based model should be viewed as a mechanism. Networks capable of operating with this attention mechanism and possessing RNN characteristics are considered attention-based. The primary goal of such a mechanism is to assign varying weights to the input sequence, allowing for the capture of data and the utilization of input-output relationships. The fundamental resolution for this architecture involves implementing a second network.

In pursuit of this objective, the authors opted for the Luong attention-based model. The weight, denoted as $w_t$, is computed for each timestep $t$ in the source during the decoding process of the attention-based encoder-decoder, with the constraint $\Sigma_s w_t(s) = 1$ and $\forall s; w_t(s) \geq 0$. The hidden state $h_t$ serves as a function representing the predicted token for the corresponding timestep, given by $\Sigma_s w_t(s) * \hat{h}_s$.

Various mathematical applications of the attention mechanism exhibit differences in how they calculate weights. In the Luong model, the computation involves applying the softmax function to the scaled scores of each token. The matrix $W_a$ linearly transforms the dot product of the decoder's $h_t$ and the encoder's $\hat{h}_s$ to obtain the score.

## B. EXTREME GRADIENT BOOSTING

Decision trees work on the principle of divide and conquer [17]. Each tree has a root node from where the checking of the selected condition starts, and on the basis of which further movement along the tree continues through internal nodes, until a leaf is reached, which actually represents the predicted class in the context of solving the classification problem. One of the main advantages of the decision tree compared to other models is the simplicity of model understanding and interpretation.

The first thing that is done when creating a tree is deciding which attribute from our data in the dataset will be the root node, based on which branching will be done. The attribute that gives the least impurity is chosen. Impurity can be calculated in several ways, such as information gain and the Gini index, which is the most used. For the attribute for which the Gini index is calculated, it is first necessary to calculate how it is related to the predicted outcome. If only one class appears for a certain attribute value, and the others do not, such a node is clean and has a gini impurity of 0. In general, the gini impurity for leaf nodes created by division by the value of a certain attribute is calculated according to the formula (1), where $p_j$ is the prior probability that a sample belongs to class j. While the total gini impurity for the attribute is calculated with a weighted sum of gini impurities of leaf nodes.

$$Gini\ impurity = 1 - \sum_j (p_j)^2 \tag{1}$$

Thus, for the root node, the attribute that gives the smallest impurity is chosen. Then it continues with the same process and creates internal nodes based on the attributes from the dataset until the leaves are reached. A leaf is created when a node gives an impurity of 0. Here, of course, one must not insist on completely clean leaves, in order not to overfit the tree.

One of the main disadvantages of the decision tree is accuracy, and as a result, many decision tree-based machine learning algorithms were created with certain modifications that solve the problem of insufficient precision. Modifications are mainly based on ensemble methods that combine several models to achieve better results. Possible improvement of the decision tree is represented by an algorithm called Random Forest [18]. According to this algorithm, many decision trees are generated, and when creating each one, only a subset of all possible attributes is considered. Finally, the classification decision is made based on the result chosen by the largest number of trees. Based on the creation of more trees, the AdaBoost [19] algorithm was also created, which creates more small trees (root node and leaves), which during their creation take into account the results of previous trees, and some trees have more influence during the final count.

Along with some analogies like the AdaBoost algorithm, there is also the Gradient Boost [20] algorithm. Gradient Boost starts building predictions starting from a single leaf that represents the prediction. Then the creation of the trees continues, which are limited in size but still larger than with AdaBoost. The tree is created based on the mistakes made before. All new trees are scaled by the learning rate parameter, which generally represents a small value between 0 and 1. Finally, the prediction is made starting from the first leaf, to which the predictions of the other scaled trees are added.

The extreme gradient boost (XGboost) algorithm is based on the idea of a gradient boost algorithm, with very high scalability. The beginning of the algorithm is the same as with Gradient Boost, with the initial prediction in one leaf. After that, the tree is built with slightly more specific rules than before. First, the root of the tree is selected as one of the attributes, and for the root of the tree, as well as for the leaves, the similarity score for classification is calculated according to the formula (2), where $\lambda$ is the regularization parameter. The regularization parameter prevents overfitting of the tree, reducing sensitivity to outliers. The minimum number of residuals in a leaf is defined with [previous probability x (1-previous probability)] which is called a cover. The default cover value is 1.

$$Similarity\ score = \frac{\sum(res_i)^2}{\sum[pprob_i \times (1 - pprob_i)] + \lambda} \tag{2}$$

in the equation $res_i$ represents the residual and $pprob_i$ previous probability,

Once the similarity scores for the nodes have been calculated, the gain for the root of the tree is calculated. The attribute with the highest gain is chosen as the root. Further, the construction of the tree continues according to the same principle up to the defined depth. When the construction of the tree is finished, tree pruning is done according to the $\lambda$ parameter. Before the predictions themselves, the output is calculated for each leaf according to the similarity score formula only without squaring the sum of residuals. The prediction for classification itself is done similarly to the Gradient Boost algorithm, starting from the initial prediction for which the log $(odds)$ is calculated, to which the result of the tree multiplied by the learning rate $lr$ parameter is added. For classification, the obtained value is put into the logistic function, to get the new prediction. To see how good the predictions are, the loss function can be calculated according to formula (3), where T is the number of leaves in the tree and w are output values of the leaves. The goal is to minimize the loss.

$$Loss = \sum_{i=1}^{n} L(y_i,\ F(x_i)) + \gamma T\ + \frac{1}{2}\lambda||w||^2 \tag{3}$$

Trees are built until the maximum depth is reached, or until the residuals become small enough.

### C. METAHEURISTIC OPTIMIZATION

For achieving the best results of machine learning algorithms, the choice of the best hyperparameters is of key importance. While the number of these hyperparameters was smaller, finding the best values was achieved by a simple trial and error method that achieved satisfactory results. However, as the number of these hyperparameters is very large today, finding the best values becomes a very difficult NP-hard problem. One of the possible solutions for hyperparameter optimization is metaheuristics.

Metaheuristics find inspiration in our environment, in nature, in physics, etc. They can be inspired by various phenomena in nature, such as animal behavior, or phenomena in physics or chemistry [21]. Perhaps one of the most well-known metaheuristics is ant colony optimization (ACO) [22]. When ants find food that they carry to their perch, on the way back they release pheromones on the way they came, so that other ants know to go the same way. This is the basic idea of the ACO algorithm. Another interesting phenomenon is swarms of animals such as birds, insects, herds, etc [23]. These swarms work together to reach a goal like finding food, and in the process learn and influence each other. The particle swarm optimization (PSO) algorithm is based on this principle. The behavior of cuckoos was also observed [24]. When laying eggs, these birds can also take other nests if that gives their eggs a better chance. Then they can even throw out host eggs and take over the nest. The algorithm based on the breeding behavior of cuckoos is called the cuckoo search algorithm.

Some of the metaheuristics such as central force optimization (CFO) found inspiration in physics. CFO bases its principles on gravitational kinematics. The expected behavior is that probes move over time and approach large masses with the largest gravitational field, which can be applied to optimization problems as well. The spiral phenomenon can also be taken as an interesting inspiration for metaheuristics [25]. The spiral optimization (SPO) algorithm performs a search using multiple spiral models, in which the search points follow the spiral trajectories toward the common center, not using a gradient. The law of gravity, as the most important postulate in physics, had to give rise to some metaheuristics. The gravitational search algorithm (GSA) is based on the principle that all particles attract each other with a force proportional to their masses and inversely proportional to the square of their distances. At the end of the GSA algorithm, objects with the largest mass will indicate the best solution.

Working with metaheuristics enabled better results in all areas, including Parkinson's disease. There are works in the field of detecting Parkinson's disease with the help of ACO [26], PSO [27] or cuckoo search [28]. In other areas, metaheuristic applications can also be found, such as CFO in node localization in wireless sensor networks [29], or GSA in the search for critical failure surface in slope stability [30]. Additional, optimization algorithms have shown great performance tackling complex tasks across several fields with hybrid algorithms further boosting performance [31], [32], [33], [34].

## III. METHODS

The metaheuristics used in this paper are described in the methods section. First, the recently introduced algorithm is described and then the modified version is presented that has been developed for this specific problem. The introduced modifications propose potential solutions for observed drawbacks of the original optimizer.

### A. THE BASIC SINH COSH OPTIMIZER

The sinh cosh optimizer (SCHO) [35] is a recently introduced optimization algorithm inspired by mathematical concepts. As a population-based metaheuristic, in the initial stages, the algorithm defines a population with a high degree of randomness as shown in Eq 4.

$$A = \begin{bmatrix} a_{1,1} \dots a_{1,j} \dots a_{1,D} \\ a_{1,2} \dots a_{2,j} \dots a_{2,D} \\ a_{N,1} \dots a_{N,j} \dots a_{N,D} \end{bmatrix} \tag{4}$$

here $P$ denotes a population of agents. Each agent's $A_{i,j}$ position is determined as per Eq 5. The values of $D$ and $N$ represent the solution dimensional space and number of agents.

$$a = rnd(N, D) \times (ub, lb) + lb \tag{5}$$

where $rnd$ denotes a random number, while $ub$ and $lb$ denote upper and lower constraints of the search space.

Following initialization, the algorithm needs to balance exploration and exploitation and guide agents towards promising seagoing of the search space. Exploration is split across tho strategies and balanced is governed by Eq 6:

$$S = floor(\frac{T}{ct}) \tag{6}$$

where $T$ denotes the maximum allocated iterations and $ct$ is a control coefficient empirically selected and set to 3.6.

During exploration agents are updated as per Eq 7:

$$A_{(i,j)}^{t+1} = \begin{cases} A_{best}^{(j)} + r_1 \times W_1 \times A_{(i,j)}^{t} & r_2 > 0.5 \\ A_{best}^{(j)} - r_1 \times W_1 \times A_{(i,j)}^{t} & r_2 < 0.5 \end{cases} \tag{7}$$

here $t$ denote the iteration number, $A_{(i,j)}^{t+1}$ describes the $j$-th and $i$-th agents and $A_{best}^{(j)}$ signifies the best agent in the $j$ dimension. Random values from a range of [0, 1] are selected for $r_1$ and $r_2$. The value of $W_1$ denotes a weighted coefficient of the specific agent and can be determined as per:

$$W_1 = r_3 \times b_1 \times (coshr_4 + \mu \times sinhr_4 - 1) \tag{8}$$

the value of the $b_1$ is decreased gradually though the iteration, and $r_3$ and $r_4$ are randomly selected values form a range of [0, 1]. A sensitivity parameter is also introduced as $\mu$.

The second strategy used during exploration applies Eq 9

$$A_{(i,j)}^{t+1} = \begin{cases} A_{best}^{(j)} + |\epsilon \times W_2 \times A_{best}^{(j)} - A_{i,j}^{(t)}| & r_5 > 0.5 \\ A_{best}^{(j)} - |\epsilon \times W_2 \times A_{best}^{(j)} - A_{i,j}^{(t)}| & r_5 < 0.5 \end{cases} \tag{9}$$

in this equation $\epsilon$ is set to 0.003 as suggested in the original work. The weight coefficient $W_2$ is determined as per:

$$W_2 = r_6 \times b_2 \tag{10}$$

here $r_6$ denote an arbitrary value in range [0, 1] and $b_2$ denotes a slowly descending value.

The other prominent stage of the optimization is exploitation during which agents focus on promising areas of the search space making more refined moments towards optima. Once again two strategies are employed by the metaheuristic. The first stage utilized Eq 11

$$A_{(i,j)}^{t+1} = \begin{cases} A_{best}^{(j)} + r_7 \times W_3 \times A_{(i,j)}^{t} & r_8 > 0.5 \\ A_{best}^{(j)} - r_7 \times W_3 \times A_{(i,j)}^{t} & r_8 < 0.5 \end{cases} \tag{11}$$

the values of $r_7$ and $r_8$ are chosen from a range of [0, 1] and $W_3$ is determined as per:

$$W_3 = r_9 \times b_1 \times (coshr_{10} + \mu \times sinhr_{10}) \tag{12}$$

here $r_9$ and $r_{10}$ represent randomly selected values from [0, 1].

The second strategy uses Eq 13

$$A_{(i,j)}^{t+1} = A_{(i,j)}^{t} + r_{11} \times \frac{singr_{12}}{coshr_1 2} |W_2 \times A_{best}^{t} - A_{i,j}^{t}| \tag{13}$$

with $r_{11}$ and $r_{12}$ values once again randomly selected from a [0, 1] range.

### B. MODIFIED SCHO OPTIMIZER

While the original SCHO has shown admirable performance compared to contemporary optimizers under CEC [36] standard evolution function testing as with many new algorithms, there is room for improvement. Two mechanisms are introduced into the basic algorithm in an attempt to boost performance.

The initial mechanism presented is derived from the ABC algorithm [37]. Depleted solutions are replaced with newly

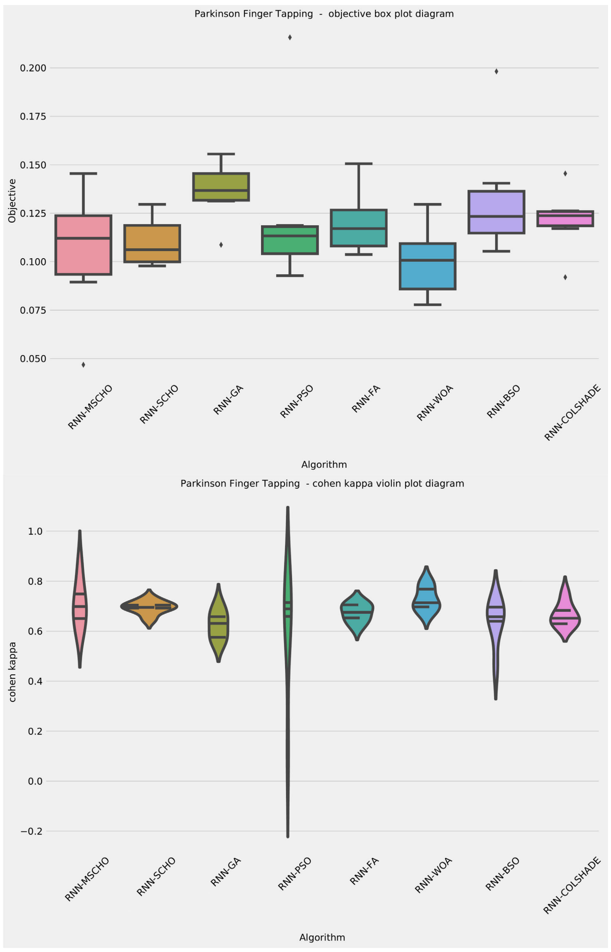

**FIGURE 1.** **Objective and indicator outcome distribution for RNN simulations.**

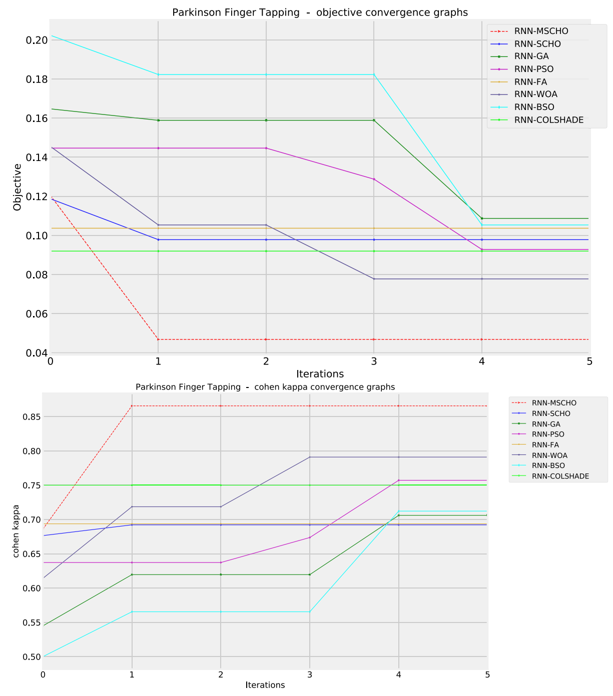

**FIGURE 2.** **Objective and indicator function convergence for RNN simulations.**

generated ones if they fail to demonstrate improvement, and within the constraints of the limited experiment iterations, solutions that do not enhance performance are rejected after two iterations.

This strategy has proven effective in enhancing exploration. The second introduced mechanism is quasi-reflective learning (QRL) [38]. This technique is employed to generate additional solutions, further amplifying the exploration process. Furthermore, during the algorithm's initialization stages, this mechanism is utilized for generating potential solutions, with the quasi-reflected component $z$ of the solution $A$ location being determined as:

$$A_z^{qr} = rad\left(\frac{lb_z + ub_z}{2}, a_z\right) \tag{14}$$

where $lb$ and $ub$ denote lower and upper bounds of the search space and $rad$ denotes a random value within the given interval. The introduced algorithm is named the modified SCHO (MSCHO). The pseudocode for the described optimizer is presented in Algorithm 1.

**Algorithm 1** Pseudocode for the Described MSCHO Algorithm

```
Set initial parameter values
Initialized population using QRL mechanism
while $T > t$ do
    Utilize appropriate SCHO strategy to update agent locations depending on $t$
    Determine agent fitness using an objective function
    for agent $p$ in Population do
        if $p$ did not improve for 2 iterations then
            Generate new solution using QRL mechanism sand replace $p$
        end if
    end for
end while
return Optimal solution from the population
```

## IV. EXPERIMENTAL SETUP

The experimental setup for this work encompasses three simulations. The first two simulations handle the time-series classification of finder-taping signals. The first experiment evaluates the performance of simple RNNs. The second experiment incorporates attention layers into the RNN in an attempt to improve outcomes. Recurrent networks with attention layers are labeled as RNN-ATT in the presented tables. Finally, the third experiment incorporates additional statistical signal analysis of patient data and handles multi-class classification of several neurodegenerative conditions adjacent to PD. The dataset used for these

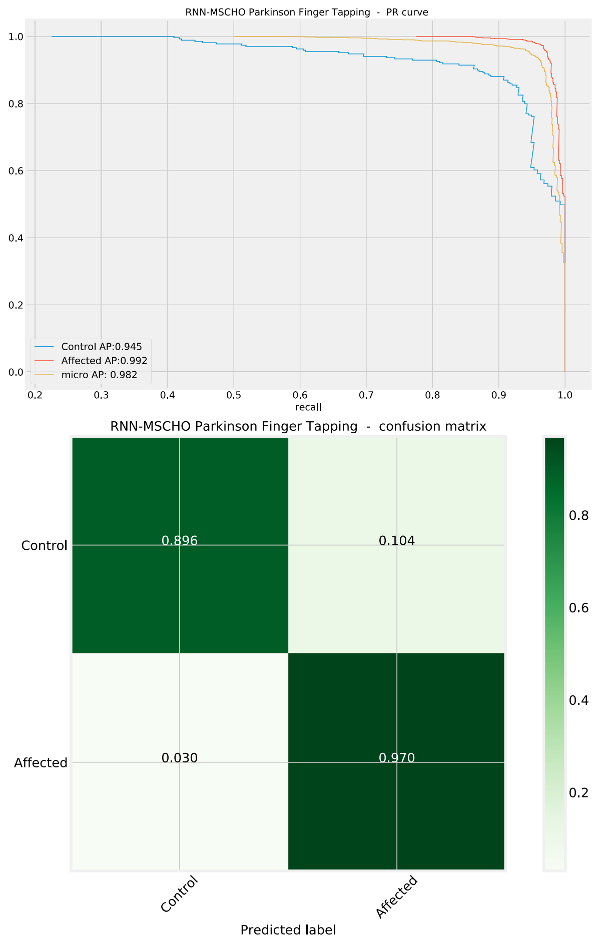

FIGURE 3. PR and confusion metrics of the best-attained model during RNN simulations.

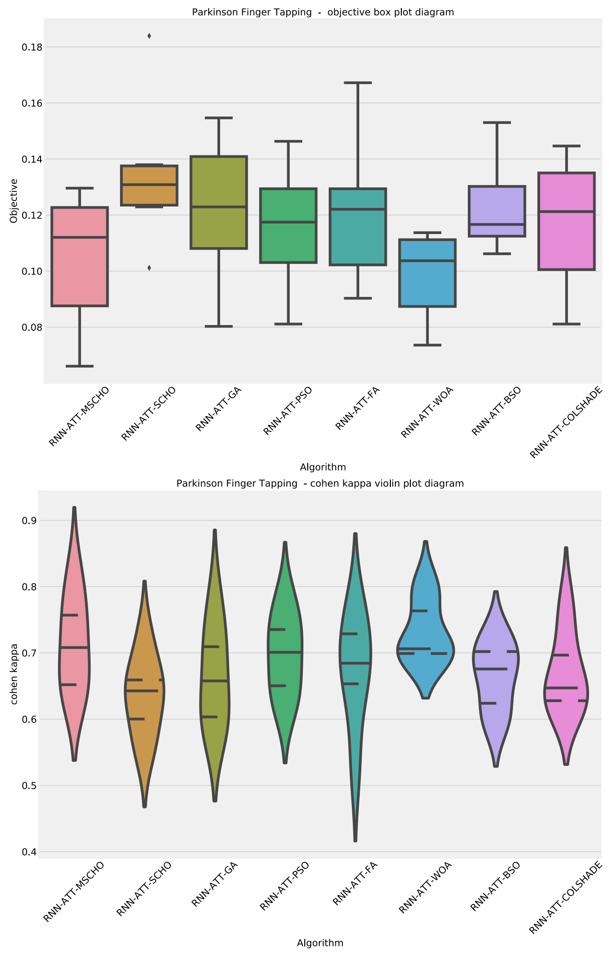

FIGURE 4. Objective and indicator outcome distribution for RNN-ATT simulations.

simulations is publicly available.[1] And consists of clinically confirmed individuals suffering from PD as well as other neurodegenerative conditions. The dataset is fairly balanced covering a diverse group of individuals.

Gyroscope data collected from several patients over multiple trials is recombined into a uniform dataset and normalized. Segments are labeled as originating from control group patients of individuals affected by the condition. Time series data is separated into the training and testing data as a 70%/30% split. Data formulated this way is used for the first two experiments with RNN and RNN-ATT models and their optimization.

The third experiment utilized per-patient data with statistical parameters including the RMS angular velocity of the x-axis of the index finger, index finger axis y angular velocity average of maxima of individual taps, the RMS index finger axis y angular acceleration, the Fourier transformation of the angular velocity vector of the index finger vector. Additionally, thumb angular velocity maximum and acceleration SD for the x and z axes respective are included. A total of six features is used for the final dataset used to evaluate XGBoost classification. The initial 70% of data is allocated for training and the later 30% for testing and evaluation.

Hyperparameters that have a high influence on RNN performance have been selected for optimization. These include architecture settings such as the number of layers between 1 and 2, and neurons in each layer [5, 15]. Additionally, dropout factors are optimized between 0.05 and 0.2, learning rate is selected form a [0.0001, 0.01] range number of training epochs from [30, 60]. In addition, the number of neurons in the attention layer in RNN-ATT networks is tuned from a range [5, 15] for the second simulation. Similarly, for the third simulation, XGboost parameters that show a significant effect on classification outcomes are tuned. These include learning rate selected from a range [0.1, 0.9], minimum child weight [1, 10], subsample [0.01, 1], colsample by tree [0.01, 1], max depth [3, 10] and gamma [0, 0.8].

As the selection of hyperparameters can be considered an NP-hard problem, for the requirements of this work, several contemporary optimizer metaheuristics have been implemented and subjected to a comparative analysis.

[1]https://github.com/innovation-center-etf/FINGER-TAPPING-PARKINSONISMS-DATABASE

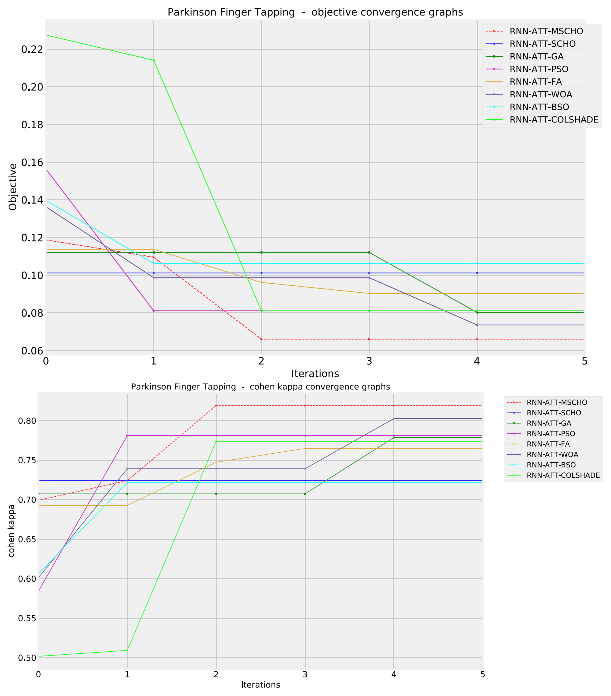

**FIGURE 5.** **Objective and indicator function convergence for RNN-ATT simulations.**

Algorithms are evaluated on their ability to select parameter values that show improved performance. Apart from the introduced modified metaheuristic, the original SCHO [35] algorithms are also evaluated. Additionally the GA [39], PSO [40], FA [41], WOA [42], BSO [43] and COLSHADE [44] algorithms are included in the comparison. These algorithms are implemented with their respective hyperparameters set to the values proposed in the original works that introduced them.

To guide the optimization process several evaluation metrics are tracked for each of the conducted simulations. Standard metrics such as accuracy, precision, recall, and f1-score are tracked [45]. All simulations used error rate as the objective function that can be determined as per:

$$Error\_rate = 1 - accuracy \tag{15}$$

An additional indicator metric is also included. Cohen's Kappa [46] is used tracked as it has shown excellent results when dealing with dissemblance data. This metric is calculated as the following:

$$\kappa = \frac{z_o - z_e}{1 - z_e} \tag{16}$$

in which $z_o$ denotes the observed and $z_e$ the ground truth values.

## V. SIMULATION OUTCOMES

The following presents the outcomes of each simulation carried out in this work.

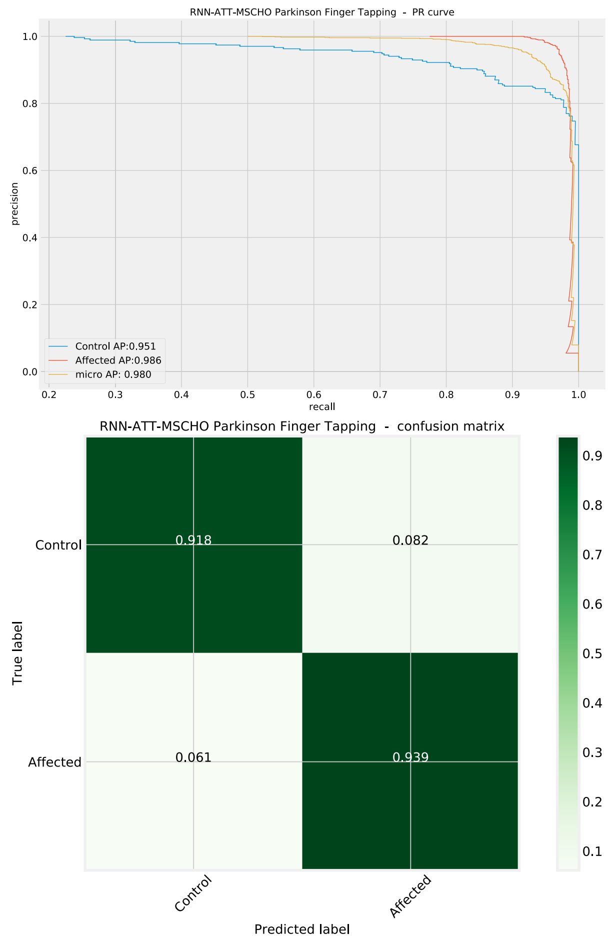

**FIGURE 6.** **PR and confusion metrics of the best-attained model during RNN-ATT simulations.**

### A. RNN SIMULATION OUTCOMES

Outcomes in terms of objective and indicator functions for the best, worst, mean as well as median runs for RNN simulations are shown alongside the standard divination, and variance are provided in Table 1 and Table 2.

From the conducted simulations with RNN, it can be deduced that the introduced algorithms generated models with bet best performing model as well as the lowest error rate in the mean average as shown in Table 1. However, admirable stability has also been shown for the original SCHO algorithm, which demonstrated the bat outcomes in the worst-case execution scenario, as well as median executions.

Indicator outcomes in Table 2 once again indicate that the introduced algorithm constructed a single best-performing model, with the base algorithm showing a high rate of stability. Nevertheless, it is worth mentioning that the WOA algorithm did attain the best outcomes in the worst mean and median runs when optimizing RNN.

Outcomes distributions of the objective and indicator functions for RNN simulations are provided in Figure 1 as a visual comparison between algorithms in terms of stability.

**TABLE 1. Objective function outcomes for RNN simulations.**

| Method | Best | Worst | Mean | Median | Std | Var |
|---|---|---|---|---|---|---|
| RNN-MSCHO | **0.046823** | 0.145485 | **0.105212** | 0.112040 | 0.031279 | 0.000978 |
| RNN-SCHO | 0.097826 | **0.129599** | 0.110089 | **0.106187** | **0.011890** | **0.000141** |
| RNN-GA | 0.108696 | 0.155518 | 0.136009 | 0.136706 | 0.014735 | 0.000217 |
| RNN-PSO | 0.092809 | 0.215719 | 0.125975 | 0.113294 | 0.041068 | 0.001687 |
| RNN-FA | 0.103679 | 0.150502 | 0.120541 | 0.117057 | 0.015823 | 0.000250 |
| RNN-WOA | 0.077759 | 0.129599 | 0.100334 | 0.100753 | 0.017485 | 0.000306 |
| RNN-BSO | 0.105351 | 0.198161 | 0.133779 | 0.123328 | 0.030793 | 0.000948 |
| RNN-COLSHADE | 0.091973 | 0.145485 | 0.121377 | 0.123746 | 0.015816 | 0.000250 |

**TABLE 2. Indicator function outcomes for RNN simulations.**

| Method | Best | Worst | Mean | Median | Std | Var |
|---|---|---|---|---|---|---|
| RNN-MSCHO | **0.865706** | 0.590296 | 0.708735 | 0.698205 | 0.088508 | 0.007834 |
| RNN-SCHO | 0.692214 | 0.648249 | 0.694210 | 0.694902 | **0.023969** | **0.000575** |
| RNN-GA | 0.706084 | 0.565419 | 0.625102 | 0.630911 | 0.053224 | 0.002833 |
| RNN-PSO | 0.757195 | 0.115095 | 0.603778 | 0.689226 | 0.220990 | 0.048836 |
| RNN-FA | 0.693745 | 0.616867 | 0.672779 | 0.674922 | 0.033927 | 0.001151 |
| RNN-WOA | 0.791036 | **0.676520** | **0.728672** | **0.713412** | 0.043491 | 0.001891 |
| RNN-BSO | 0.712264 | 0.458101 | 0.638016 | 0.657515 | 0.085128 | 0.007247 |
| RNN-COLSHADE | 0.750058 | 0.626136 | 0.665753 | 0.651375 | 0.043904 | 0.001928 |

**TABLE 3. Detailed comparison between best-performing models attained during RNN simulations.**

| Method | Metric | Control | PD | Accuracy | Macro avg | Weighted avg |
|---|---|---|---|---|---|---|
| RNN-MSCHO | precision | **0.895911** | 0.969795 | **0.953177** | **0.9328529** | **0.953177** |
| | recall | 0.895911 | 0.969795 | **0.953177** | **0.9328529** | **0.953177** |
| | f1-score | **0.895911** | **0.969795** | **0.953177** | **0.9328529** | **0.953177** |
| RNN-SCHO | precision | 0.876238 | 0.907445 | 0.902174 | 0.891841 | 0.900426 |
| | recall | 0.657993 | **0.973031** | 0.902174 | 0.815513 | 0.902174 |
| | f1-score | 0.751592 | 0.939094 | 0.902174 | 0.845343 | 0.896922 |
| RNN-GA | precision | 0.720635 | 0.952327 | 0.891304 | 0.836481 | 0.900216 |
| | recall | 0.843866 | 0.905070 | 0.891304 | 0.874468 | 0.891304 |
| | f1-score | 0.777397 | 0.928097 | 0.891304 | 0.852747 | 0.894202 |
| RNN-PSO | precision | 0.730994 | **0.977752** | 0.907191 | 0.854373 | 0.922252 |
| | recall | **0.929368** | 0.900755 | 0.907191 | 0.915062 | 0.907191 |
| | f1-score | 0.818331 | 0.937675 | 0.907191 | 0.878003 | 0.910833 |
| RNN-FA | precision | 0.793522 | 0.923077 | 0.896321 | 0.858300 | 0.893938 |
| | recall | 0.728625 | 0.944984 | 0.896321 | 0.836804 | 0.896321 |
| | f1-score | 0.759690 | 0.933902 | 0.896321 | 0.846796 | 0.894719 |
| RNN-WOA | precision | 0.775000 | 0.976027 | 0.922241 | 0.875514 | 0.930813 |
| | recall | 0.921933 | 0.922330 | 0.922241 | 0.922132 | 0.922241 |
| | f1-score | 0.842105 | 0.948419 | 0.922241 | 0.895262 | 0.924508 |
| RNN-BSO | precision | 0.732899 | 0.950506 | 0.894649 | 0.841703 | 0.901563 |
| | recall | 0.836431 | 0.911543 | 0.894649 | 0.873987 | 0.894649 |
| | f1-score | 0.781250 | 0.930617 | 0.894649 | 0.855933 | 0.897022 |
| RNN-COLSHADE | precision | 0.755627 | 0.961582 | 0.908027 | 0.858604 | 0.915259 |
| | recall | 0.873606 | 0.918015 | 0.908027 | 0.895811 | 0.908027 |
| | f1-score | 0.810345 | 0.939294 | 0.908027 | 0.874819 | 0.910291 |
| | support | 269 | 927 | | | |

Convergence rates of the objective and indicator functions for RNN simulations are provided in Figure 2 for visual comparisons.

A clear improvement in convergence rate can be observed in the introduced algorithm, which has demonstrated an ability to overcome local optima traps and coverage toward more promising solutions.

Detailed metrics comparisons between the best-performing models attained by each optimizer during RNN summations are provided in Table 3.

A clear advantage can be observed for the models optimized by the introduced algorithm, which have showcased the highest accuracy as well as admirable results across most test cases.

Graphics of the PR curves and the confusion matrix of the best-performing RNN model are shown in Figure 3.

To facilitate experimental repeatability parameter selections made by each algorithm for the respective best-performing models during RNN simulations are provided in Table 4.

### B. RNN-ATT SIMULATION OUTCOMES

Outcomes in terms of objective and indicator functions for the best, worst, mean as well as median runs for RNN-ATT

**TABLE 4.** Parameter selection for the best-attained model during RNN simulations by each algorithm.

| Method | Learning Rate | Dropout | Epochs | Layers | Neurons L1 | Neurons L2 |
|---|---|---|---|---|---|---|
| RNN-MSCHO | 0.008453 | 0.147989 | 60 | 2 | 7 | 12 |
| RNN-SCHO | 0.010000 | 0.200000 | 46 | 2 | 15 | 10 |
| RNN-GA | 0.007574 | 0.200000 | 56 | 1 | 15 | N/a |
| RNN-PSO | 0.006854 | 0.200000 | 52 | 1 | 15 | N/a |
| RNN-FA | 0.010000 | 0.200000 | 60 | 2 | 5 | 10 |
| RNN-WOA | 0.010000 | 0.200000 | 60 | 2 | 15 | 15 |
| RNN-BSO | 0.006139 | 0.059525 | 43 | 2 | 13 | 9 |
| RNN-COLSHADE | 0.006295 | 0.200000 | 55 | 2 | 15 | 15 |

**TABLE 5.** Objective function outcomes for RNN-ATT simulations.

| Method | Best | Worst | Mean | Median | Std | Var |
|---|---|---|---|---|---|---|
| RNN-ATT-MSCHO | **0.066054** | 0.129599 | 0.104236 | 0.112040 | 0.023215 | 0.000539 |
| RNN-ATT-SCHO | 0.101171 | 0.183946 | 0.134615 | 0.130853 | 0.025125 | 0.000631 |
| RNN-ATT-GA | 0.080268 | 0.154682 | 0.121795 | 0.122910 | 0.024853 | 0.000618 |
| RNN-ATT-PSO | 0.081104 | 0.146321 | 0.115663 | 0.117475 | 0.021304 | 0.000454 |
| RNN-ATT-FA | 0.090301 | 0.167224 | 0.121516 | 0.122074 | 0.025193 | 0.000635 |
| RNN-ATT-WOA | 0.073579 | **0.113712** | **0.098384** | **0.103680** | **0.015125** | **0.000229** |
| RNN-ATT-BSO | 0.106187 | 0.153010 | 0.123049 | 0.116639 | 0.016014 | 0.000256 |
| RNN-ATT-COLSHADE | 0.081104 | 0.144649 | 0.116918 | 0.121237 | 0.022593 | 0.000510 |

**TABLE 6.** Indicator function outcomes for RNN-ATT simulations.

| Method | Best | Worst | Mean | Median | Std | Var |
|---|---|---|---|---|---|---|
| RNN-ATT-MSCHO | **0.818912** | 0.638314 | 0.713302 | **0.708058** | 0.065759 | 0.004324 |
| RNN-ATT-SCHO | 0.724029 | 0.551677 | 0.635280 | 0.642669 | 0.055014 | 0.003027 |
| RNN-ATT-GA | 0.778547 | 0.583289 | 0.664825 | 0.657689 | 0.069863 | 0.004881 |
| RNN-ATT-PSO | 0.780966 | 0.640558 | 0.697136 | 0.700855 | 0.056171 | 0.003155 |
| RNN-ATT-FA | 0.764607 | 0.531504 | 0.675209 | 0.684350 | 0.075428 | 0.005689 |
| RNN-ATT-WOA | 0.802515 | **0.701431** | **0.731894** | 0.706202 | **0.042949** | **0.001845** |
| RNN-ATT-BSO | 0.721526 | 0.599700 | 0.665009 | 0.675682 | 0.046389 | 0.002152 |
| RNN-ATT-COLSHADE | 0.773655 | 0.623387 | 0.669840 | 0.647078 | 0.055641 | 0.003096 |

**TABLE 7.** Detailed comparison between best-performing models attained during RNN-ATT simulations.

| Method | Metric | Control | PD | Accuracy | Macro avg | Weighted avg |
|---|---|---|---|---|---|---|
| RNN-ATT-MSCHO | precision | **0.812500** | 0.975336 | **0.933946** | **0.893918** | **0.938712** |
| | recall | 0.918216 | **0.938511** | **0.933946** | 0.928363 | **0.933946** |
| | f1-score | **0.862129** | **0.956570** | **0.933946** | **0.909349** | **0.935328** |
| RNN-ATT-SCHO | precision | 0.740260 | 0.953829 | 0.898829 | 0.847044 | 0.905794 |
| | recall | 0.847584 | 0.913700 | 0.898829 | 0.880642 | 0.898829 |
| | f1-score | 0.790295 | 0.933333 | 0.898829 | 0.861814 | 0.901162 |
| RNN-ATT-GA | precision | 0.789298 | 0.963211 | 0.919732 | 0.876254 | 0.924095 |
| | recall | 0.877323 | 0.932039 | 0.919732 | 0.904681 | 0.919732 |
| | f1-score | 0.830986 | 0.947368 | 0.919732 | 0.889177 | 0.921192 |
| RNN-ATT-PSO | precision | 0.772152 | 0.971591 | 0.918896 | 0.871871 | 0.926734 |
| | recall | 0.907063 | 0.922330 | 0.918896 | 0.914697 | 0.918896 |
| | f1-score | 0.834188 | 0.946320 | 0.918896 | 0.890254 | 0.921100 |
| RNN-ATT-FA | precision | 0.733333 | **0.981199** | 0.909699 | 0.857266 | 0.925450 |
| | recall | **0.940520** | 0.900755 | 0.909699 | 0.920638 | 0.909699 |
| | f1-score | 0.824104 | 0.939258 | 0.909699 | 0.881681 | 0.913358 |
| RNN-ATT-WOA | precision | 0.781931 | 0.979429 | 0.926421 | 0.880680 | 0.935008 |
| | recall | 0.933086 | 0.924488 | 0.926421 | **0.928787** | 0.926421 |
| | f1-score | 0.850847 | 0.951165 | 0.926421 | 0.901006 | 0.928602 |
| RNN-ATT-BSO | precision | 0.708824 | 0.967290 | 0.893813 | 0.838057 | 0.909156 |
| | recall | 0.895911 | 0.893204 | 0.893813 | 0.894557 | 0.893813 |
| | f1-score | 0.791461 | 0.928772 | 0.893813 | 0.860117 | 0.897888 |
| RNN-ATT-COLSHADE | precision | 0.796552 | 0.958057 | 0.918896 | 0.877305 | 0.921732 |
| | recall | 0.858736 | 0.936354 | 0.918896 | 0.897545 | 0.918896 |
| | f1-score | 0.826476 | 0.947081 | 0.918896 | 0.886779 | 0.919955 |
| | support | 269 | 927 | | | |

simulations are shown alongside the standard divination and variance are provided in Table 5 and Table 6.

From the conducted simulations with RNN-ATT, it can be deduced that the introduced algorithms generated models

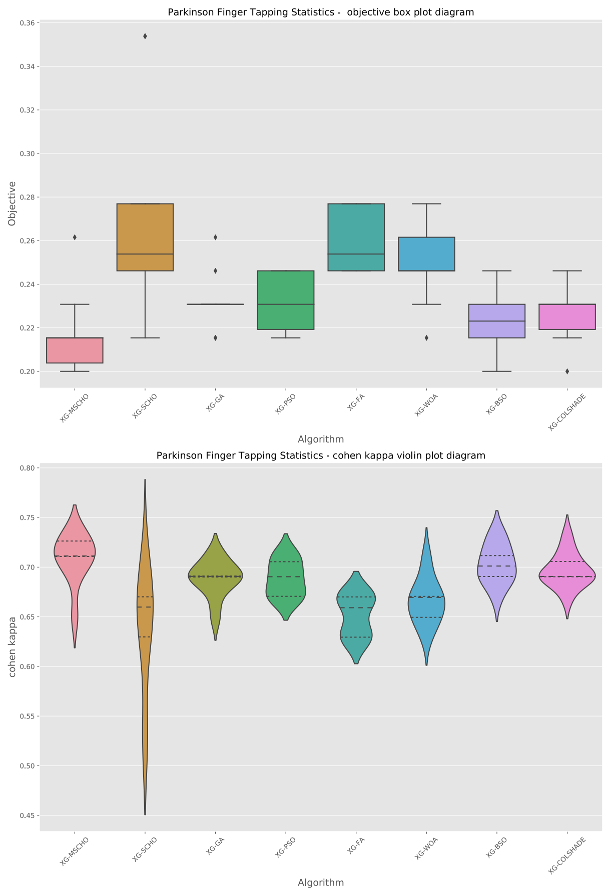

**FIGURE 7. Objective and indicator outcome distribution for XGBoost simulations.**

with bet best-performing model as shown in Table 5. However, admirable outcomes have also been shown for the original WOA algorithm. This is to be expected as per the "No free lunch" (NFL) [47] theorem, no single approach is equally suited to all challenges according to all metrics. Constant experimentation is required to determine the best method of problem pairings.

The introduced algorithm optimized the best model in terms of indicator functions as well as shown in Table 6 as well as in terms of median outcomes. However, the WOA once again showcase decent outcomes in term of worst and median outcomes.

Outcomes distributions of the objective and indicator functions for RNN-ATT simulations are provided in Figure 4 as a visual comparison between algorithms in terms of stability.

Distribution graphs suggest that the WOA shows great promise in terms of model stability.

Convergence rates of the objective and indicator functions for RNN-ATT simulations are provided in Figure 5 for visual comparisons.

Convergence rates indicated that the introduced modification allows the modified algorithm to overcome local minimum solutions, and focus towards finding a better solution in the search space.

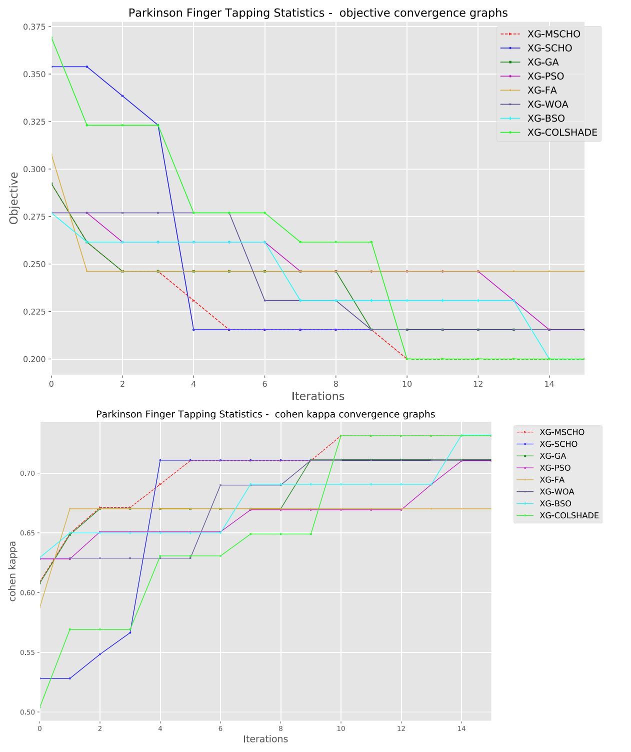

**FIGURE 8. Objective and indicator function convergence for XGBoost simulations.**

Detailed metrics comparisons between the best-performing models attained by each optimizer during RNN-ATT summations are provided in Table 7.

Detailed metrics indicate that the best performing model optimized by the introduced metaheuristic attained the best outcomes across multiple metrics and indicated the best accuracy as shown in Table 7.

Graphics of the PR curves and the confusion matrix of the best-performing RNN-ATT model are shown in Figure 6.

To facilitate experimental repeatability parameter selections made by each algorithm for the respective best-performing models during RNN-ATT simulations are provided in Table 8.

### C. XGBOOST SIMULATION OUTCOMES

Outcomes in terms of objective and indicator functions for the best, worst, mean as well as median runs for XGboost simulations are shown alongside the standard divination and variance are provided in Table 9 and Table 10.

In cases of the objective function, the introduced optimizer attained the best outcomes or matched the best outcomes presented by another optimizer, however, all algorithms performed fairly well comparatively.

In terms of the indicator function, the introduced optimizer also attained the best outcomes in the best case while the PSO demonstrated the highest stability.

Outcomes distributions of the objective and indicator functions for XGBoost simulations are provided in Figure 7 as a visual comparison between algorithms in terms of stability.

**TABLE 8.** Parameter selection for the best-attained model during RNN-ATT simulations by each algorithm.

| Method | Learning Rate | Dropout | Epochs | Layers | Neurons L1 | Neurons L2 | ATT Neurons |
|---|---|---|---|---|---|---|---|
| RNN-ATT-MSCHO | 0.010000 | 0.200000 | 60 | 2 | 15 | 15 | 7 |
| RNN-ATT-SCHO | 0.006555 | 0.166548 | 59 | 1 | 14 | N/a | 11 |
| RNN-ATT-GA | 0.009202 | 0.116195 | 59 | 1 | 15 | N/a | 8 |
| RNN-ATT-PSO | 0.004887 | 0.122132 | 58 | 2 | 15 | 15 | 15 |
| RNN-ATT-FA | 0.010000 | 0.200000 | 60 | 2 | 15 | 15 | 15 |
| RNN-ATT-WOA | 0.010000 | 0.200000 | 60 | 2 | 14 | 15 | 14 |
| RNN-ATT-BSO | 0.010000 | 0.122906 | 60 | 2 | 10 | 12 | 15 |
| RNN-ATT-COLSHADE | 0.010000 | 0.121180 | 60 | 1 | 15 | N/a | 9 |

**TABLE 9.** Objective function outcomes for XGBoost simulations.

| Method | Best | Worst | Mean | Median | Std | Var |
|---|---|---|---|---|---|---|
| XG-MSCHO | 0.200000 | 0.261538 | **0.216923** | **0.215385** | 0.017474 | 0.000305 |
| XG-SCHO | 0.215385 | 0.353846 | 0.272308 | 0.253846 | 0.044082 | 0.001943 |
| XG-GA | 0.215385 | 0.261538 | 0.232308 | 0.230769 | 0.012779 | 0.000163 |
| XG-PSO | 0.215385 | 0.246154 | 0.232308 | 0.230769 | 0.012779 | 0.000163 |
| XG-FA | 0.246154 | 0.276923 | 0.260000 | 0.253846 | 0.014514 | 0.000211 |
| XG-WOA | 0.215385 | 0.276923 | 0.249231 | 0.246154 | 0.016570 | 0.000275 |
| XG-BSO | 0.200000 | 0.246154 | 0.221538 | 0.223077 | 0.014100 | 0.000199 |
| XG-COLSHADE | 0.200000 | 0.246154 | 0.226154 | 0.230769 | 0.012016 | 0.000144 |

**TABLE 10.** Indicator function outcomes for XGBoost simulations.

| Method | Best | Worst | Mean | Median | Std | Var |
|---|---|---|---|---|---|---|
| XG-MSCHO | **0.731831** | 0.649651 | **0.709054** | **0.711294** | 0.023219 | 0.000539 |
| XG-SCHO | 0.710836 | 0.528093 | 0.635876 | 0.659808 | 0.058200 | 0.003387 |
| XG-GA | 0.711203 | 0.648983 | 0.688544 | 0.690722 | 0.017081 | 0.000292 |
| XG-PSO | 0.710283 | 0.669001 | 0.688425 | 0.690378 | **0.016907** | **0.000286** |
| XG-FA | 0.670155 | 0.630098 | 0.651514 | 0.659143 | 0.019230 | 0.000370 |
| XG-WOA | 0.710744 | 0.630215 | 0.665948 | 0.669527 | 0.021867 | 0.000478 |
| XG-BSO | 0.731320 | **0.670260** | 0.703106 | 0.701134 | 0.018842 | 0.000355 |
| XG-COLSHADE | 0.731320 | 0.669421 | 0.696564 | 0.690624 | 0.016067 | 0.000258 |

**TABLE 11.** Detailed comparison between best-performing models attained during XGBoost simulations.

| Method | Metric | 0 | 1 | 2 | 3 | Accuracy | Macro avg | Weighted avg |
|---|---|---|---|---|---|---|---|---|
| XG-MSCHO | precision | 0.916667 | 0.764706 | 0.888889 | 0.666667 | 0.800000 | 0.809232 | 0.798089 |
| | recall | 0.916667 | 0.722222 | 0.941176 | 0.666667 | 0.800000 | 0.811683 | 0.800000 |
| | f1-score | 0.916667 | 0.742857 | 0.914286 | 0.666667 | 0.800000 | 0.810119 | 0.798681 |
| XG-SCHO | precision | 0.750000 | 0.842105 | 0.750000 | **0.785714** | 0.784615 | 0.781954 | 0.785396 |
| | recall | 0.750000 | **0.888889** | 0.882353 | 0.611111 | 0.784615 | 0.783088 | 0.784615 |
| | f1-score | 0.750000 | **0.864865** | 0.810811 | **0.687500** | 0.784615 | 0.778294 | 0.780405 |
| XG-GA | precision | 0.846154 | 0.722222 | 0.888889 | 0.687500 | 0.784615 | 0.786191 | 0.779076 |
| | recall | 0.916667 | 0.722222 | 0.941176 | 0.611111 | 0.784615 | 0.797794 | 0.784615 |
| | f1-score | 0.880000 | 0.722222 | 0.914286 | 0.647059 | 0.784615 | 0.790892 | 0.780768 |
| XG-PSO | precision | 0.909091 | **0.789474** | 0.800000 | 0.666667 | 0.784615 | 0.791308 | 0.780302 |
| | recall | 0.833333 | 0.833333 | 0.941176 | 0.555556 | 0.784615 | 0.790850 | 0.784615 |
| | f1-score | 0.869565 | 0.810811 | 0.864865 | 0.606061 | 0.784615 | 0.787825 | 0.779095 |
| XG-FA | precision | 0.846154 | 0.733333 | 0.800000 | 0.647059 | 0.753846 | 0.756637 | 0.747706 |
| | recall | 0.916667 | 0.611111 | 0.941176 | 0.611111 | 0.753846 | 0.770016 | 0.753846 |
| | f1-score | 0.880000 | 0.666667 | 0.864865 | 0.628571 | 0.753846 | 0.760026 | 0.747338 |
| XG-WOA | precision | 0.916667 | 0.722222 | 0.842105 | 0.687500 | 0.784615 | 0.792124 | 0.779858 |
| | recall | 0.916667 | 0.722222 | 0.941176 | 0.611111 | 0.784615 | 0.797794 | 0.784615 |
| | f1-score | 0.916667 | 0.722222 | 0.888889 | 0.647059 | 0.784615 | 0.793709 | 0.780895 |
| XG-BSO | precision | 0.846154 | 0.764706 | 0.888888 | 0.705882 | 0.800000 | 0.801408 | 0.795931 |
| | recall | 0.916667 | 0.722222 | 0.941176 | 0.666667 | 0.800000 | 0.811683 | 0.800000 |
| | f1-score | 0.880000 | 0.742857 | 0.914286 | 0.685714 | 0.800000 | 0.805714 | 0.797187 |
| XG-COLSHADE | precision | 0.916667 | 0.764706 | 0.888889 | 0.666667 | 0.800000 | 0.809232 | 0.798089 |
| | recall | 0.916667 | 0.722222 | 0.941176 | 0.666667 | 0.800000 | 0.811683 | 0.800000 |
| | f1-score | 0.916667 | 0.742857 | 0.914286 | 0.666667 | 0.800000 | 0.810119 | 0.798681 |
| | support | 12 | 18 | 17 | 18 | | | |

Convergence rates of the objective and indicator functions for XGBoost simulations are provided in Figure 8 for visual comparisons.

Convergence rate improvements can once again be confirmed by graphs shown in Table 8.

Detailed metrics comparisons between the best-performing models attained by each optimizer during XGBoost summations are provided in Table 11.

Graphics of the PR curves and the confusion matrix of the best-performing XGBoost model are shown in Figure 9.

**TABLE 12. Paramater selection for the best-attained model during XGBoost simulations by each algorithm.**

| Method | Learning Rate | Min child weight | Subsample | Colsample by tree | Max Depth | Gamma |
|---|---|---|---|---|---|---|
| XG-MSCHO | 0.282075 | 1.000000 | 0.900722 | 1.000000 | 10 | 0.000000 |
| XG-SCHO | 0.293307 | 1.000000 | 1.000000 | 0.759122 | 10 | 0.268169 |
| XG-GA | 0.100903 | 1.000000 | 0.987141 | 1.000000 | 10 | 0.800000 |
| XG-PSO | 0.812386 | 1.016655 | 1.000000 | 0.810564 | 8 | 0.765072 |
| XG-FA | 0.900000 | 1.000000 | 1.000000 | 1.000000 | 4 | 0.800000 |
| XG-WOA | 0.594996 | 1.000000 | 1.000000 | 1.000000 | 10 | 0.343158 |
| XG-BSO | 0.104486 | 1.000000 | 0.991391 | 1.000000 | 5 | 0.243472 |
| XG-COLSHADE | 0.723806 | 1.093269 | 1.000000 | 1.000000 | 5 | 0.347276 |

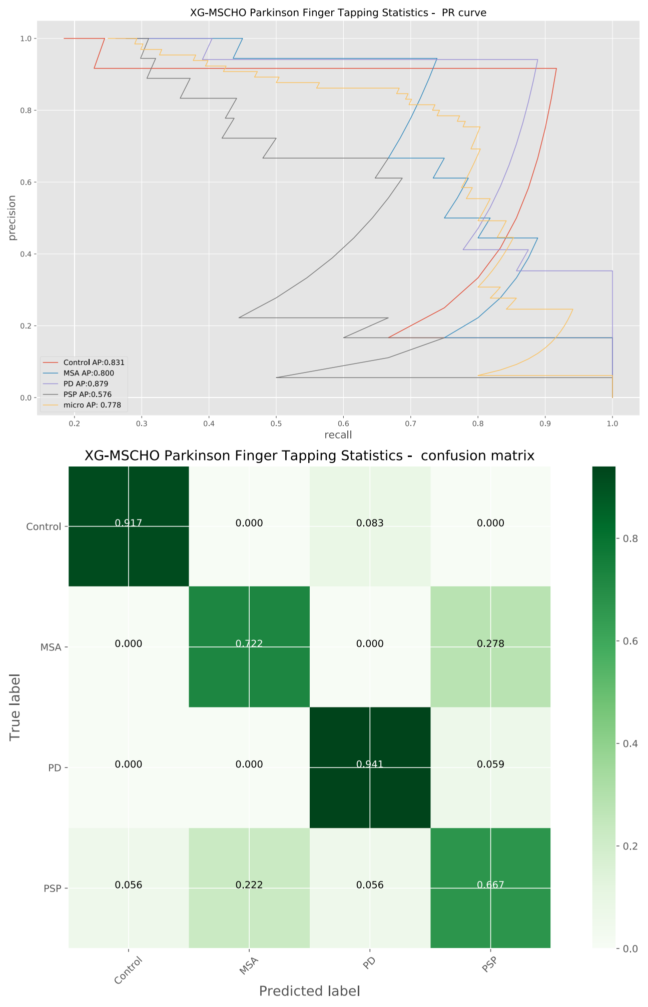

**FIGURE 9. PR and confusion metrics of the best-attained model during XGBoost simulations.**

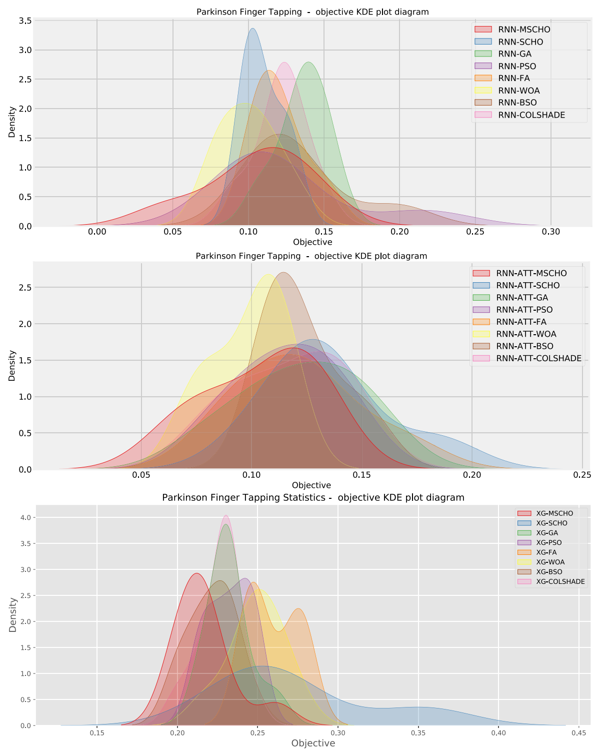

**FIGURE 10. KDE plots for the objective function outcomes in each conducted experiment.**

To facilitate experimental repeatability parameter selections made by each algorithm for the respective best-performing models during XGBoost simulations are provided in Table 12.

## D. OUTCOME STATISTICAL VALIDATION

Exploring challenges in optimization underscores the importance of model assessment. It is crucial to comprehend the statistical significance of any enhancements introduced. Relying solely on outcomes is insufficient for establishing the superiority of one algorithm over another.

Previous studies indicate that a thorough statistical evaluation should occur after sampling the examined methodologies comprehensively. This involves computing objective averages across numerous independent runs. Moreover, it is essential for samples to conform to a normal distribution to prevent drawing inaccurate inferences. The utilization of objective function averages for comparing stochastic algorithms remains a topic of debate among researchers [48].

In establishing the statistical significance of observed outcomes, samples were generated using the optimal values from 30 independent executions of each metaheuristic. Nevertheless, it was imperative to affirm the reliability of parametric testing. To address this, the independence, normality, and homoscedasticity of data variances were taken into account, following the recommendations of [49].

The independence criterion is fulfilled by initializing each run with a pseudo-random number. However, the assumption of normality is not satisfied, as illustrated by the KED plots and corroborated by the Shapiro-Wilk test outcomes for single-problem analyses [50].

**TABLE 13.** Shapiro-wilk scores for the single-problem analysis for testing normality condition.

| Experiment | MSCHO | SCHO | GA | PSO | FA | WOA | BSO | COLSHADE |
|---|---|---|---|---|---|---|---|---|
| RNN | 0.035 | 0.023 | 0.022 | 0.026 | 0.027 | 0.030 | 0.017 | 0.014 |
| RNN-ATT | 0.035 | 0.032 | 0.037 | 0.019 | 0.022 | 0.025 | 0.037 | 0.033 |
| XGBoost | 0.029 | 0.020 | 0.025 | 0.036 | 0.015 | 0.019 | 0.026 | 0.024 |

**TABLE 14.** Wilcoxon signed-rank test findings.

| MSCHO vs. others | SCHO | GA | PSO | FA | WOA | BSO | COLSHADE |
|---|---|---|---|---|---|---|---|
| RNN | 0.035 | 0.046 | 0.036 | 0.042 | 0.043 | 0.029 | 0.040 |
| RNN-ATT | 0.041 | 0.044 | 0.046 | 0.035 | 0.024 | 0.039 | 0.037 |
| XGBoost | 0.024 | 0.043 | 0.039 | 0.048 | 0.045 | 0.044 | 0.038 |

By performing the Shapiro-Wilk test, *p*-values are generated for each method-problem combination, and these outcomes are presented in Table 13.

The standard significance thresholds of $\alpha = 0.05$ and $\alpha = 0.1$ suggest the potential rejection of the null hypothesis ($H_0$), indicating that none of the samples (across all problem-method combinations) conform to a normal distribution. This highlights that the prerequisite of normality, crucial for the dependable utilization of parametric tests, was not fulfilled, leading to the conclusion that there was no need to examine variance homogeneity.

As the requirements for the reliable application of parametric tests were not met, non-parametric tests were employed for the statistical analysis. Specifically, the Wilcoxon signed-rank test, which is a non-parametric statistical test [51], was performed on the MSCHO method and all other techniques for all three problem instances (experiments). The same data samples used in the previous normality test (Shapiro-Wilk) were used for each method. The results of this analysis are presented in Table 14, where *p*-values greater than the significance level of $\alpha = 0.05$ are highlighted in bold.

Table 14 illustrates the *p*-values obtained from the Wilcoxon signed-rank test, revealing that the proposed MSCHO approach outperformed all other strategies across all three studies.

For all other techniques, the p-values were below 0.05. Consequently, in these computationally intensive simulations, the MSCHO approach showcased both robustness and effectiveness as an optimizer. According to the statistical analysis, the MSCHO technique outperformed the majority of the other tested metaheuristics in all four studies.

## VI. CONCLUSION

This work tackles the complex and pressing issue of neurodegenerative condition detection through the application of emerging AI algorithms. Both time series classification using RNNs and attention-based RNNs are employed as well as statistical analysis with the XGBoost algorithm for early detection of neurological conditions for a real-world clinical dataset. Due to a heavy reliance on algorithms for proper hyperparameter selection needed to attain desired outcomes, metaheuristic algorithms are deployed for parameter selection and adjustment. Additionally, a modified version of the recently proposed SCHO algorithm is proposed and employed in this study to select parameters in a set of three experiments. The introduced optimizer shows potential couples with RNN, attention-based RNN, and XGBoost, with the best-constructed models exceeding 90% accuracy for time-series classification and 80% on simple classifications.

Certain limitations exist within this study. Limited amounts of data are available which constrains the amount of available training data. Furthermore, extensive computational demands limit population sizes that can be tested for metaheuristic optimizes. Additionally, only a small subset of well-established algorithms are included in the comparative analysis.

Future works hope to further improve the methodology for early detection of neurological disorders through optimization. Other algorithms should be explored such as ash echo-state networks. Furthermore, the potential of the introduced modified optimizer will be explored for other challenging tasks form the medical field.

## REFERENCES

[1] L. V. Kalia and A. E. Lang, “Parkinson’s disease,” *Lancet*, vol. 386, no. 9996, pp. 896–912, Aug. 2015.

[2] N. R. McFarland, “Diagnostic approach to atypical parkinsonian syndromes,” *CONTINUUM, Lifelong Learn. Neurol.*, vol. 22, no. 4, pp. 1117–1142, 2016.

[3] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, Jul. 2015.

[4] P. C. Sen, M. Hajra, and M. Ghosh, “Supervised classification algorithms in machine learning: A survey and review,” in *Emerging Technology in Modelling and Graphics*. Singapore: Springer, 2018, pp. 99–111.

[5] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, and M. Tuba, “Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics,” *Algorithms*, vol. 13, no. 3, p. 67, Mar. 2020.

[6] S. S. V. Chandra and A. H. S. Anand, “Nature inspired meta heuristic algorithms for optimization problems,” *Computing*, vol. 104, no. 2, pp. 251–269, Feb. 2022.

[7] S. Chakraborty, S. Aich, and H.-C. Kim, “Detection of Parkinson’s disease from 3T T1 weighted MRI scans using 3D convolutional neural network,” *Diagnostics*, vol. 10, no. 6, p. 402, Jun. 2020.

[8] S. Shinde, S. Prasad, Y. Saboo, R. Kaushick, J. Saini, P. K. Pal, and M. Ingalhalikar, “Predictive markers for Parkinson’s disease using deep neural nets on neuromelanin sensitive MRI,” *NeuroImage, Clin.*, vol. 22, Mar. 2019, Art. no. 101748.

[9] I. El Maachi, G.-A. Bilodeau, and W. Bouachir, “Deep 1D-convnet for accurate Parkinson disease detection and severity prediction from gait,” *Expert Syst. Appl.*, vol. 143, Apr. 2020, Art. no. 113075.

[10] E. Abdulhay, N. Arunkumar, K. Narasimhan, E. Vellaiappan, and V. Venkatraman, “Gait and tremor investigation using machine learning techniques for the diagnosis of Parkinson disease,” *Future Gener. Comput. Syst.*, vol. 83, pp. 366–373, Jun. 2018.

[11] M. Yokoe, R. Okuno, T. Hamasaki, Y. Kurachi, K. Akazawa, and S. Sakoda, “Opening velocity, a novel parameter, for finger tapping test in patients with Parkinson’s disease,” *Parkinsonism Rel. Disorders*, vol. 15, no. 6, pp. 440–444, Jul. 2009.

[12] B. Deore and S. Bhosale, "Intrusion detection system based on RNN classifier for feature reduction," *Social Netw. Comput. Sci.*, vol. 3, no. 2, p. 114, Mar. 2022.

[13] Y. Yan, L. Qi, J. Wang, Y. Lin, and L. Chen, "A network intrusion detection method based on stacked autoencoder and LSTM," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[14] G. Abdurrahman and M. Sintawati, "Implementation of Xgboost for classification of Parkinson's disease," *J. Phys., Conf. Ser.*, vol. 1538, no. 1, May 2020, Art. no. 012024.

[15] M. M. Nishat, T. Hasan, S. M. Nasrullah, F. Faisal, M. A. Asif, and M. A. Hoque, "Detection of Parkinson's disease by employing boosting algorithms," in *Proc. Joint 10th Int. Conf. Informat., Electron. Vis. (ICIEV) 5th Int. Conf. Imag., Vis. Pattern Recognit. (icIVPR)*, Aug. 2021, pp. 1–7.

[16] S.-C. Wang and S.-C. Wang, "Artificial neural network," in *Interdisciplinary Computing in Java Programming*. Springer, 2003, pp. 81–100.

[17] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *J. Chemometrics*, vol. 18, no. 6, pp. 275–285, Jun. 2004.

[18] G. Biau and E. Scornet, "A random forest guided tour," *TEST*, vol. 25, no. 2, pp. 197–227, Jun. 2016.

[19] R. E. Schapire, "Explaining AdaBoost," in *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*. Berlin, Germany: Springer, 2013, pp. 37–52.

[20] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artif. Intell. Rev.*, vol. 54, no. 3, pp. 1937–1967, Mar. 2021.

[21] O. M. Alyasiri, Y.-N. Cheah, A. K. Abasi, and O. M. Al-Janabi, "Wrapper and hybrid feature selection methods using metaheuristic algorithms for English text classification: A systematic review," *IEEE Access*, vol. 10, pp. 39833–39852, 2022.

[22] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[23] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, Jan. 2018.

[24] X. S. Yang and S. Deb, "Engineering optimisation by Cuckoo Search," *Int. J. Math. Model. Numer. Optim.*, vol. 1, no. 4, p. 330, 2010.

[25] K. Tamura and K. Yasuda, "The spiral optimization algorithm: Convergence conditions and settings," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 1, pp. 360–375, Jan. 2020.

[26] A. Ul Haq, J. Li, M. H. Memon, Z. Ali, S. Z. Abbas, and S. Nazir, "Recognition of the Parkinson's disease using a hybrid feature selection approach," *J. Intell. Fuzzy Syst.*, vol. 39, no. 1, pp. 1319–1339, 2020.

[27] D. Wu, K. Warwick, Z. Ma, M. N. Gasson, J. G. Burgess, S. Pan, and T. Z. Aziz, "Prediction of Parkinson's disease tremor onset using a radial basis function neural network based on particle swarm optimization," *Int. J. Neural Syst.*, vol. 20, no. 2, pp. 109–116, 2010.

[28] B. Sabeena, S. Sivakumari, and D. M. Teressa, "Optimization-based ensemble feature selection algorithm and deep learning classifier for Parkinson's disease," *J. Healthcare Eng.*, vol. 2022, pp. 1–12, Apr. 2022.

[29] J. S. Pan, P. C. Song, S. C. Chu, and T. Y. Wu, "An adaptive stochastic central force optimisation algorithm for node localisation in wireless sensor networks," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 39, nos. 1–2, pp. 1–19, 2022.

[30] A. W. Mohamed, K. M. Sallam, P. Agrawal, A. A. Hadi, and A. K. Mohamed, "Evaluating the performance of meta-heuristic algorithms on CEC 2021 benchmark problems," *Neural Comput. Appl.*, vol. 35, no. 2, pp. 1493–1517, Jan. 2023.

[31] L. Jovanovic, D. Jovanovic, N. Bacanin, A. J. Stakic, M. Antonijevic, H. Magd, R. Thirumalaisamy, and M. Zivkovic, "Multi-step crude oil price prediction based on LSTM approach tuned by salp swarm algorithm with disputation operator," *Sustainability*, vol. 14, no. 21, p. 14616, Nov. 2022.

[32] M. Salb, L. Jovanovic, M. Zivkovic, E. Tuba, A. Elsadai, and N. Bacanin, "Training logistic regression model by enhanced moth flame optimizer for spam email classification," in *Proc. 5th Comput. Netw. Inventive Commun. Technol. (ICCNCT)*. Singapore: Springer, 2022, pp. 753–768.

[33] N. Bacanin, L. Jovanovic, M. Zivkovic, V. Kandasamy, M. Antonijevic, M. Deveci, and I. Strumberger, "Multivariate energy forecasting via metaheuristic tuned long-short term memory and gated recurrent unit neural networks," *Inf. Sci.*, vol. 642, Sep. 2023, Art. no. 119122.

[34] L. Jovanovic, G. Jovanovic, M. Perisic, F. Alimpic, S. Stanisic, N. Bacanin, M. Zivkovic, and A. Stojic, "The explainable potential of coupling metaheuristics-optimized-XGBoost and SHAP in revealing VOCs' environmental fate," *Atmosphere*, vol. 14, no. 1, p. 109, Jan. 2023.

[35] J. Bai, Y. Li, M. Zheng, S. Khatir, B. Benaissa, L. Abualigah, and M. A. Wahab, "A Sinh Cosh optimizer," *Knowl.-Based Syst.*, vol. 282, Dec. 2023, Art. no. 111081.

[36] S. Jiang, S. Yang, X. Yao, K. C. Tan, M. Kaiser, and N. Krasnogor, "Benchmark functions for the CEC'2018 competition on dynamic multiobjective optimization," School Comput. Sci. Inform., Newcastle Univ., 2018.

[37] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, Jan. 2008.

[38] N. Bacanin, N. Budimirovic, K. Venkatachalam, H. S. Jassim, M. Zivkovic, S. S. Askar, and M. Abouhawwash, "Quasi-reflection learning arithmetic optimization algorithm firefly search for feature selection," *Heliyon*, vol. 9, no. 4, Apr. 2023, Art. no. e15378.

[39] S. Mirjalili, "Genetic algorithm," in *Evolutionary Algorithms and Neural Networks: Theory and Applications*. Berlin, Germany: Springer, 2019, pp. 43–55.

[40] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE ICNN*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[41] X. S. Yang and X. He, "Firefly algorithm: Recent advances and applications," *Int. J. Swarm Intell.*, vol. 1, no. 1, pp. 36–50, 2013.

[42] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.

[43] Y. Shi, "Brain storm optimization algorithm," in *Proc. 2nd Int. Conf. Adv. Swarm Intell. (ICSI)*, Chongqing, China. Cham, Switzerland: Springer, Jun. 2011, pp. 303–309.

[44] J. Gurrola-Ramos, A. Hernàndez-Aguirre, and O. Dalmau-Cedeño, "COLSHADE for real-world single-objective constrained optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.

[45] Ž. Đ. Vujovic, "Classification model evaluation metrics," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 6, pp. 599–606, 2021.

[46] M. J. Warrens, "Five ways to look at Cohen's kappa," *J. Psychol. Psychotherapy*, vol. 5, no. 4, pp. 1–5, 2015.

[47] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[48] T. Eftimov, P. Korošec, and B. K. Seljak, "A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics," *Inf. Sci.*, vol. 417, pp. 186–215, Nov. 2017.

[49] A. LaTorre, D. Molina, E. Osaba, J. Poyatos, J. Del Ser, and F. Herrera, "A prescription of methodological guidelines for comparing bio-inspired optimization algorithms," *Swarm Evol. Comput.*, vol. 67, Dec. 2021, Art. no. 100973.

[50] S. S. Shapiro and R. S. Francia, "An approximate analysis of variance test for normality," *J. Amer. Stat. Assoc.*, vol. 67, no. 337, p. 215, Mar. 1972.

[51] S. M. Taheri and G. Hesamian, "A generalization of the Wilcoxon signed-rank test and its applications," *Stat. Papers*, vol. 54, no. 2, pp. 457–470, May 2013.

**JELICA CINCOVIC** was born in Belgrade, in December 1994. She received the Graduate and master's degrees in software engineering from the School of Electrical Engineering, University of Belgrade, in 2017 and 2019, respectively, where she is currently pursuing the Ph.D. degree. She was a Holder of the "Vuk Karadžić" Diploma from the Third Belgrade High School, in 2013. During her studies, she completed two professional internships in the company's Asseco SEE and Avisto Eastern Europe. She defended the Graduate and master's thesis. From 2017 to 2018, she was a Teaching Assistant with the Department of Computer Engineering, School of Electrical Engineering, University of Belgrade. During her work with the University of Belgrade, she wrote several scientific and professional articles in the field of internet programming and machine learning, participated in scientific projects, and has been participating as a Lecturer with the UNDP Program for IT retraining for several years in a row.

**LUKA JOVANOVIC** received the degree from the Faculty of Technical Sciences, Singidunum University. Throughout his studies, he actively contributed to various artificial intelligence research projects, resulting in over 50 peer-reviewed publications in esteemed journals and international conferences. His research interests include reinforcement learning, natural language processing, distributed computing, and signal processing. In his current capacity with the Faculty of Technical Sciences, Singidunum University, he plays a crucial role in the realm of research. His responsibilities include spearheading research initiatives, publishing scholarly articles, conceptualizing innovative ideas, and implementing experiments through programming. This dynamic role not only allows him to apply his academic knowledge with precision but also serves as a cornerstone for his continual professional growth.

**BOSKO NIKOLIC** received the Ph.D. degree in electrical and computer engineering from the School of Electrical Engineering, University of Belgrade, Belgrade, Serbia. He is currently a Professor with the Department for Computer Science and Information Technology, School of Electrical Engineering, University of Belgrade. His research interests include information systems, artificial intelligence, machine learning, natural language processing, and the development of educational systems.

**NEBOJSA BACANIN** received the Ph.D. degree from the Faculty of Mathematics, University of Belgrade, in 2015 (study Computer Science Program). The external Committee Member of the Ph.D. thesis defense was one of the greatest world scholars from the domain of artificial intelligence, specifically metaheuristics optimization (creator of a dozen state-of-the-art swarm intelligence algorithms), Prof. Xin-She Yang, from Middlesex University, London, U.K. He started university career in Serbia 18 years ago with the Graduate School of Computer Science, Belgrade. He is currently a Full Professor, the Head of the Study Program Applied Artificial Intelligence, and a Vice-Rector of Scientific Research with Singidunum University, Belgrade, Serbia. He is involved in scientific research in the field of computer science and his specialties includes stochastic optimization algorithms, swarm intelligence, machine learning, soft computing, optimization and modelling, image processing, and cloud and distributed computing. He has published more than 320 scientific papers (more than 130 papers indexed in Clarivate Analytics SCIE) in high quality journals, articles as the book chapters and international conferences indexed in Clarivate Analytics JCR, Scopus, WoS, IEEExplore, and other scientific databases. He has also published four books from domains of cloud computing, web programming, and advanced java spring programming. He is a member of numerous editorial boards, scientific and advisory committees of international conferences and journals, and a regular reviewer of international journals with high Clarivate analytics and WoS impact factor. He also serves as an associate editor and as a guest editor for many outstanding international journals indexed in Clarivate Analytics SCIE. He has also been included in the prestigious Stanford University list with 2% best world researchers from the domain of artificial intelligence, that encompasses whole career, in 2021 and 2022, while according to the AD Scientific Index (Alper-Doger Scientific Index), he is also ranked as the second best researcher in Serbia from the domain of computer science.