



Article

# Improved Colony Predation Algorithm Optimized Convolutional Neural Networks for Electrocardiogram Signal Classification

Xinxin He <sup>1</sup>, Weifeng Shan <sup>1,\*</sup>, Ruilei Zhang <sup>1</sup>, Ali Asghar Heidari <sup>2</sup>, Huiling Chen <sup>3,\*</sup>  
and Yudong Zhang <sup>4,\*</sup>

<sup>1</sup> School of Emergency Management, Institute of Disaster Prevention, Sanhe 065201, China; hexinxin0305@gmail.com (X.H.); zhangrl420@163.com (R.Z.)

<sup>2</sup> School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran 1417935840, Iran; as\_heidari@ut.ac.ir

<sup>3</sup> Institute of Big Data and Information Technology, Wenzhou University, Wenzhou 325000, China

<sup>4</sup> School of Computing and Mathematical Sciences, University of Leicester, Leicester LE1 7RH, UK

\* Correspondence: william.shan@gmail.com (W.S.); chenhuiling.jlu@gmail.com (H.C.); yudongzhang@ieee.org (Y.Z.)

**Abstract:** Recently, swarm intelligence algorithms have received much attention because of their flexibility for solving complex problems in the real world. Recently, a new algorithm called the colony predation algorithm (CPA) has been proposed, taking inspiration from the predatory habits of groups in nature. However, CPA suffers from poor exploratory ability and cannot always escape solutions known as local optima. Therefore, to improve the global search capability of CPA, an improved variant (OLCPA) incorporating an orthogonal learning strategy is proposed in this paper. Then, considering the fact that the swarm intelligence algorithm can go beyond the local optimum and find the global optimum solution, a novel OLCPA-CNN model is proposed, which uses the OLCPA algorithm to tune the parameters of the convolutional neural network. To verify the performance of OLCPA, comparison experiments are designed to compare with other traditional metaheuristics and advanced algorithms on IEEE CEC 2017 benchmark functions. The experimental results show that OLCPA ranks first in performance compared to the other algorithms. Additionally, the OLCPA-CNN model achieves high accuracy rates of 97.7% and 97.8% in classifying the MIT-BIH Arrhythmia and European ST-T datasets.

**Keywords:** colony predation algorithm; convolutional neural networks; ECG; hyperparameter optimization; orthogonal learning strategy; swarm intelligence algorithm



**Citation:** He, X.; Shan, W.; Zhang, R.; Heidari, A.A.; Chen, H.; Zhang, Y. Improved Colony Predation Algorithm Optimized Convolutional Neural Networks for Electrocardiogram Signal Classification. *Biomimetics* **2023**, *8*, 268. <https://doi.org/10.3390/biomimetics8030268>

Academic Editor: Yongquan Zhou

Received: 13 May 2023

Revised: 18 June 2023

Accepted: 18 June 2023

Published: 21 June 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In recent years, research has shown that deep learning offers numerous benefits compared to conventional machine learning approaches [1,2]. Deep learning is known to extract features efficiently compared to traditional machine learning approaches, so many researchers tend to work with deep learning [3,4]. These methods are robust against noise, can achieve superior accuracy, and can be scaled straightforwardly to larger datasets, consequently reducing training time [5–7]. The main methods commonly used for deep learning are deep neural networks and generative adversarial networks. Among these networks, convolutional neural networks (CNNs) have made many contributions to the field of computer vision and are popular among researchers [8]. LeNet-5 [9] is the pioneer CNN, a convolutional neural network algorithm proposed by Yann LeCun in 1998 to solve the problem of handwriting recognition. Then, the AlexNet [10] network structure was introduced, and it won the 2012 ImageNet competition. Since then, CNNs have received widespread and enthusiastic attention worldwide, and more new network structures have

been proposed, such as VGG [11], GoogleLeNet [12], ResNet [13], and DenseNet [14]. When these new network structures were created, the number of layers and parameters increased accordingly. However, tuning hyperparameters is a highly challenging task. Since the number of parameters is very large and specialized personnel are required to select the optimal parameters based on experience, this can result in the loss of a large volume of labor resources and the waste of material resources. Therefore, it is a tough task to manually tune parameters using manual labor with limited resources.

Optimization techniques range from multiobjective methods to single objective approaches to many objective techniques [15–17]. Each of these approaches has its own unique set of computational difficulties, making the process of optimization both challenging and rewarding [18,19]. The challenges of the optimization methods presented in the previous literature include a high computational cost, difficulty in tackling multiple local optima, lack of robustness, immature convergence, and conditionality of global optima results [20–22]. As one of the main classes of optimization methods, activity patterns of various groups of organisms are used to generate swarm intelligence (SI) algorithms [23]. In recent years, SIs have been able to solve complex optimization problems in the real world due to their excellent optimization capabilities [24,25]. Some of the famous algorithms are the particle swarm algorithm (PSO) [26], Harris hawk optimization algorithm (HHO) [27], slime mould algorithm (SMA) [28,29], hunger games search (HGS) [30], Runge Kutta optimizer (RUN) [31], the weighted mean of vectors (INFO) [32], colony predation algorithm (CPA) [33], and rime optimization algorithm (RIME) [34]. They have been applied to solve many problems, such as bankruptcy prediction [35], economic emission dispatch [36], feature selection [37,38], constrained multiobjective optimization [39], dynamic multiobjective optimization [40], global optimization [41], medical image segmentation [42], feed-forward neural networks [43], scheduling optimization [44], large-scale complex optimization [45], multiobjective optimization [46], and numerical optimization [47]. Among them, CPA is a new algorithm proposed in recent years that is based on prey predation by animal groups in nature. CPA has a more vital optimization ability than PSO, MFO, and other algorithms [33]. Because swarm intelligence algorithms can solve complex practical problems, many researchers have proposed optimizing the parameters of deep learning network structures using SI. Researchers also need to assess the performance of swarm intelligence algorithms. IEEE CEC2017 serves as a benchmark function for testing the performance of such algorithms and comprises four categories: unimodal functions, multimodal functions, hybrid functions, and composition functions.

There are currently two types of researchers studying deep learning network structures. Some researchers use manually configured deep learning network structures, while others use hyperparameters that use SIs to optimize deep learning network structures. Pyakillya et al. [48] proposed a model for automatic classifications using a deep learning architecture that consists of a one-dimensional convolutional layer and a fully connected layer. The model achieved an accuracy of 86% on the validation dataset. Mathews et al. [49] proposed a deep learning model for electrocardiogram (ECG) classification that incorporates a restricted Boltzmann machine and a deep belief network. The experiments showed that this model performed better at low sampling rates. Sannino et al. [50] proposed a deep neural network with seven hidden layers for automatic classification and verified experimentally that this model outperforms other models in terms of accuracy, achieving a precision of 99.52%. Strodthoff et al. [51] proposed a deep learning-based time series classification algorithm that mainly uses the ResNet network model, and the experimental results proved that the performance of this algorithm is promising. Peimankar et al. [52] proposed a method for ECG signal detection composed of deep learning-based convolutional neural networks and long- and short-term memory. Different heartbeat waveforms were detected from the MITDB and QTDB datasets to test the method's effectiveness. The F1 scores obtained on the two datasets are 99.56% and 96.78%, respectively, indicating that this method was very effective for ECG signal detection. Hasan et al. [53] proposed the use of one-dimensional convolutional neural networks for the recognition of multiple heart

diseases, and the accuracy of this method on the MIT-BIH, St-Petersberg, and PTB datasets is 97.7%, 99.71%, and 98.24%, respectively. Acharya et al. [54] investigated a nine-layer convolutional neural network structure for identifying five different classes of heartbeats in ECG signals, and the experimental results showed that the accuracy was 94.03%.

Compared with manual search, automatic search using SI can be performed by improving the algorithm to find the values of suitable parameters within the selected range and finally find the most optimal values.

In the study of Houssein et al. [55], a convolutional neural network model based on an improved marine predators algorithm (IMPA-CNN) was proposed to exploit the best classification role of CNNs and to find the best hyperparameters of CNNs. This model uses the improved marine predators algorithm to select the most suitable CNN parameters automatically, and the experimental results of testing it on different ECG datasets show that this model is very effective. Khalifa et al. [56] proposed a method to optimize the parameters of a seven-layer CNN network; the first six layers of the CNN use gradient descent and the last layer uses a particle swarm algorithm to find the optimal parameters of the CNN. This model was compared with a handwritten character recognition dataset using a standard CNN architecture, and the results show that this model has a higher accuracy, with a value of 96.67%. Yamasaki et al. [57] proposed a method to improve image recognition accuracy, in which a particle swarm algorithm is implemented to automatically find the appropriate hyperparameters of the CNN. The results show higher accuracy when using this method to optimize the AlexNet structure and compare it with the standard AlexNet-CNN structure in image recognition experiments. Dey et al. [58] proposed a model integrating three network structures, VGG19, ResNet50, and DenseNet121, used to screen tuberculosis or TB images from chest X-rays. To overcome the problem of manual tuning, the training part of the model uses an optimization algorithm to set the parameters of the model, and this method was proven effective for TB classification by testing on a TB dataset.

Using convolutional neural networks, Pathan et al. [59] investigated a method to automatically identify chest X-ray images affected by COVID-19. This method achieved 98.8% and 96% accuracy for dataset 1 and dataset 2, respectively. The results of the experiments on COVID-19 images show that this method can effectively screen out patients with the disease. The hyperparameters of the DenseNet121 architecture were optimized using the gravity search algorithm in the work of Ezzat et al. [60]. This optimized model was compared with the CNN architecture of Inception-v3 in experiments for the detection of new crown pneumonia, and the experimental results indicate that this model performs exceptionally well in terms of accuracy, reaching 98.38%, significantly higher than its competitor. Most studies used optimization algorithms to tune the parameters of CNNs, while some works used optimization algorithms to tune the overall architecture of CNNs to select the most appropriate number of network layers. In Singh et al. [61]'s study, a multi-stage particle swarm was used to optimize the network structure and hyperparameters of CNNs, which is divided into two stages. In the first stage, the multi-level particle swarm algorithm is used to optimize the CNN architecture and determine the number of network layers that can better exploit the performance of the CNN. In the second stage, the hyperparameters are tuned based on this network structure. The final model was tested using an image dataset, and the results show that the performance of this model is excellent. To speed up finding the layers and parameters of CNN architecture, Fernandes et al. [62] proposed a new particle swarm velocity operator and used this new particle swarm algorithm to optimize the architecture and parameters of a CNN. Experimental tests show that this model can find an optimized CNN model based on any dataset.

Although many researchers have studied this area, there are still many challenges to be tackled. CPA faces the same challenges as other swarm intelligence algorithms, such as falling into local optima and slow convergence. To solve these problems, CPA needs to be improved. Therefore, this motivates us to propose an improved CPA and use it to optimize the parameters of a CNN.

This paper proposes an improved CPA based on an orthogonal learning strategy (OLCPA). To demonstrate the effectiveness of OLCPA in optimizing CNN parameters, it was applied to the classification of arrhythmia in ECG datasets. The main contributions of this paper are as follows:

- An OLCPA algorithm based on the orthogonal learning strategy is proposed, and it is compared with four traditional and seven advanced algorithms on the IEEE CEC 2017 benchmark functions.
- This paper analyzes the scalability of OLCPA and CPA on different dimensions of the IEEE CEC2017 benchmark functions.
- A CNN-based OLCPA-CNN model for identifying abnormal ECG signals is designed.
- The OLCPA-CNN model is compared with other methods using the MIT-BIH and the European ST-T datasets.

The rest of this paper is as follows: Section 2 describes CPA and introduces the background knowledge on CNNs. Section 3 describes the improved CPA algorithm (OLCPA). The process of the OLCPA-CNN model is described in Section 4. Section 5 describes the experimental design and results of OLCPA. Section 6 describes the application of OLCPA-CNN. Section 7 is the discussion. Finally, Section 8 is the conclusion of this paper.

## 2. Preliminary Work

### 2.1. Overview of Colony Predation Algorithm

The colony predation algorithm [33] was inspired by the fact that cooperative communicative group predation of group-housed animals increases the probability of successful predation.

Group-living animals pursue their prey by communicating with each other, and Equation (1) simulates this process.

$$X_j^i(t + 1) = X_j^i(t) + (1 - r) \cdot \left( \frac{X_1(t) + X_2(t)}{2} \right) \tag{1}$$

where  $X_j^i(t)$  is the individual currently searching for prey in the  $j$ -th dimension,  $j$  ranges from 1 to  $\text{dim}$ , and  $i$  represents the current individual.  $X_1$  and  $X_2$  are the positions of the two individuals closest to the prey,  $r$  is a random number between 0 and 1, and  $X_j^i(t + 1)$  is the position of the individual in the next iteration.

Two strategies are used in the pursuit process to increase the probability of successful predation: scattering prey and surrounding prey. Prey dispersal is to drive the prey in different directions and weaken the prey group, and Equation (2) simulates this process.

$$X(t + 1) = X_{best} - S \cdot (r_1 \cdot (ub - lb) + lb) \tag{2}$$

where  $S$  denotes the energy of the prey and its value is changed,  $lb$  and  $ub$  are the left and right values of the boundary range,  $r_1$  is a random number between 0 and 1,  $X_{best}$  is the location of the prey, and  $X(t + 1)$  is the current position of the pursuer. The variation of  $S$  is as follows:

$$S_0 = a - t \cdot \left( \frac{a}{N} \right) \tag{3}$$

$$S = 2 \cdot S_0 \cdot r_2 \tag{4}$$

$$a = e^{w - 2w(1 - \frac{t}{MaxFes})} \tag{5}$$

where  $r_2$  varies between 0 and 1,  $t$  is the current number of evaluations, and  $N$  indicates the number of predators.  $S_0$  varies with the number of evaluations, the value of  $a$  is related to the number of evaluations, and the value of  $w$  is 9.

After the prey has been successfully dispersed, the predators use a siege attack on them, a process shown in Equation (6):

$$X(t + 1) = X_{best} - 2S \cdot D \cdot e^l \cdot \tan\left(\frac{\pi}{4}l\right) \tag{6}$$

$$D = |X_{best} - X(t)| \tag{7}$$

where  $D$  is the interval indicating the current individual's distance from the prey. The probability of the two strategies being executed is shown in Equation (8).

$$X(t + 1) = \begin{cases} X_{best} - S \cdot (r_1 \cdot (ub - lb) + lb) & r \geq 0.5 \\ X_{best} - 2S \cdot D \cdot e^l \cdot \tan\left(\frac{\pi}{4}l\right) & r < 0.5 \end{cases} \tag{8}$$

When the predator begins to chase prey, there are two strategies: one is when the predators find prey nearby, and choose to support the closest predator to the prey; Equation (9) simulates this process. The second is when the predators do not find prey around them, they will randomly choose other locations of prey; this process is shown in Equation (11).

$$X(t + 1) = P_{nearest} \tag{9}$$

$$D1 = |2r_4 \cdot X_{rand} - X(t)| \tag{10}$$

$$X(t + 1) = X_{rand} - S \cdot D1 \tag{11}$$

$$X_{rand} = r_5 \cdot ((ub - lb) + lb) \tag{12}$$

where  $P_{nearest}$  denotes the position of the individual closest to the prey,  $D1$  denotes the distance moved by the random population, and  $X_{rand}$  denotes the new position randomly generated by the prey individuals.

The probability of the above two strategies being executed is described in Equation (13). Algorithm 1 describes the process of implementing CPA.

$$X(t + 1) = \begin{cases} P_{nearest} & |r_6| \leq 1 \\ X_{rand} - S \cdot D1 & |r_6| > 1 \end{cases} \tag{13}$$

---

**Algorithm 1** Pseudo-code for CPA

---

Initialize population size  $Num$ , the problem dimension  $dim$ , and the maximum number of evaluations  $MaxFes$

**While** ( $t \leq MaxFes$ )

**For**  $i = 1: Num$

Calculation of individual fitness values

Update  $X_{best}$

**End for**

**For**  $j = 1: dim$

Update  $X_1, X_2$

Calculate  $X_j^i$  using Equation (1)

**End for**

**For**  $i = 1: Num$

Update  $S$

**If**  $|S| < \frac{2}{3}a$

Calculate the current agent's position by Equation (8)

**End if**

**If**  $|S| \geq \frac{2}{3}a$

Calculate the current agent's position by Equation (13)

**End if**

**End for**

$t = t + 1$

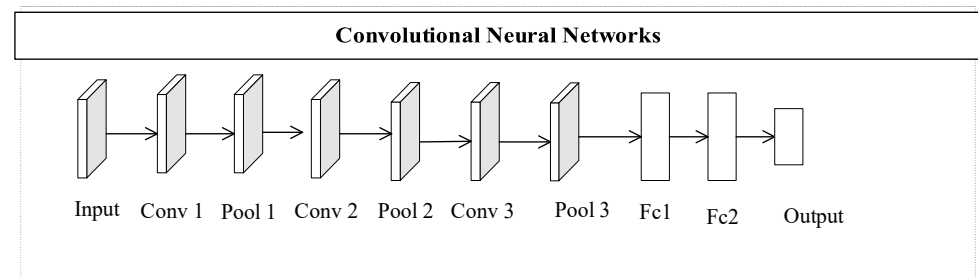
**End while**

Return  $X_{best}$

---

## 2.2. Convolutional Neural Network

Yann Lecun of New York University proposed the convolutional neural network in 1998. It differs from a multilayer perceptron (MLP) and is used in various fields, such as image processing. The difference with regard to MLP is that the way of CNN local connection and weight sharing is changed, where on the one hand, the network can be better optimized by reducing the number of weights, and on the other hand, the complexity of the model can be effectively reduced. The convolutional neural network structure is a deep neural network with a convolutional structure, and its overall architecture of network structure includes an input layer, convolutional layer, rectified linear units (ReLU) layer, pooling layer, and fully connected layer. In practical applications, the convolutional layer contains the convolutional layer and the ReLU layer. Activation functions are usually used to compute the convolutional layer, and the commonly used activation functions are the Sigmoid, Tanh, and ReLU functions. The pooling layer is generally arranged after the convolutional layer to reduce the network's parameters and computational resources. The role of the fully connected layer in the entire network is to classify, and it is usually found in the last few layers of the CNN. The one-dimensional CNN structure of this paper is shown in Figure 1.



**Figure 1.** The architecture of a nine-layer CNN.

## 3. The Improved CPA

CPA is an algorithm with better optimization performance. However, when faced with complex optimization problems, it tends to fall into local optima or slow down the convergence rate due to the fact that it lacks some strategies that can flexibly address these problems. To overcome these problems, we propose an improved CPA that incorporates an orthogonal learning strategy.

### 3.1. Orthogonal Learning Design

An orthogonal learning design [63] is a method that uses a small number of experiments to find the best solution. The determination of the small number of experiments is mainly related to two factors in the orthogonal table  $L_M$ : the  $K$  factor and the number of  $Q$  levels for each factor.  $L_M(Q^K)$  indicates that  $Q^K$  sets of experiments need to be carried out, but when the values of  $Q$  and  $K$  are large, it is impossible to complete this many experiments, so it is necessary to use the orthogonal table to design the number of experiments. For experiments designed using the orthogonal table, only  $M$  combinations need to be chosen to complete the experiments, and the number of experiments  $M$  is much smaller than  $Q^K$ . For example, the following orthogonal table  $L_9(3^3)$  explains the process.



$$L_9(3^3) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 3 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{bmatrix} \tag{14}$$

In Equation (14),  $L_9$  indicates that the experiment with the orthogonal table design needs to be executed only 9 times, but without the orthogonal design, this experiment needs to be executed 27 times. Therefore, the orthogonal design can significantly reduce the number of experiments, and this method is more effective when the values of  $Q$  and  $K$  are larger.

### 3.2. Orthogonal Learning Strategy

This study introduces the orthogonal learning strategy (OL) into CPA, which uses orthogonal tables to generate a new search mechanism to explore more regions and avoid becoming trapped in local optima. After using this strategy, the original CPA generates  $M + 1$  search agents, which can improve the exploration ability of CPA.

### 3.3. The Proposed OLCPA

This subsection proposes a novel CPA algorithm based on an orthogonal learning strategy. Equation (15) describes the formation process of OLCPA. This new OLCPA algorithm exploits the orthogonal strategy’s search mechanism to expand the solution’s search scope and find high-quality solutions. The pseudo-code for OLCPA is described in Algorithm 2, and Figure 2 describes the specific process of OLCPA.

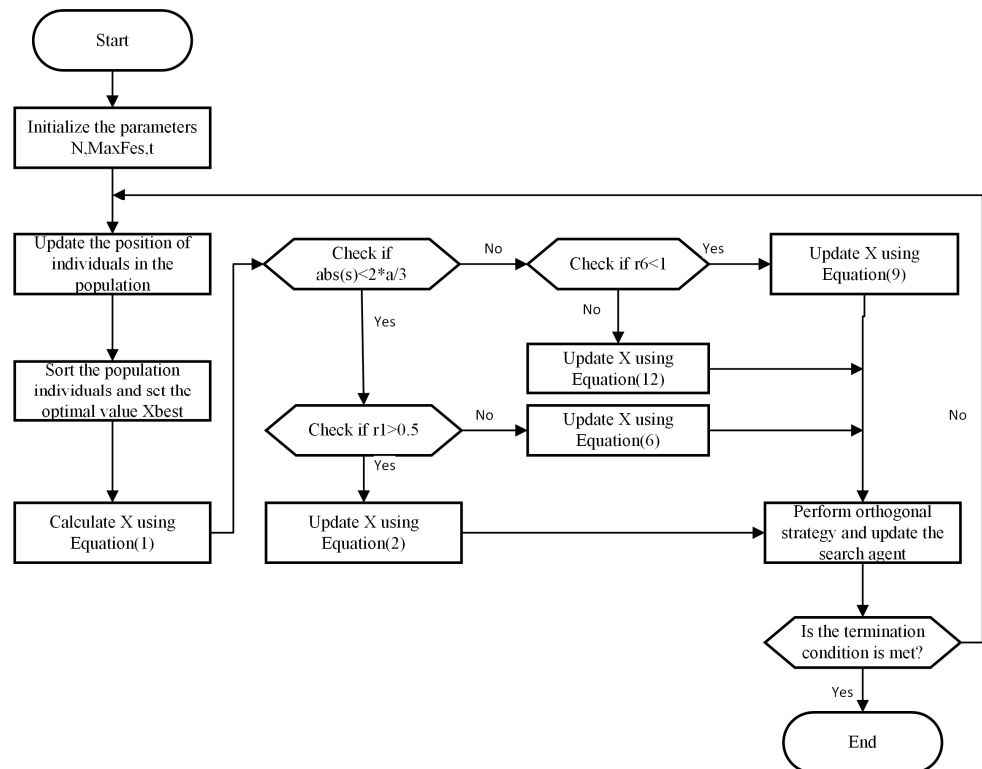


Figure 2. Flowchart of OLCPA.

$$X(t+1) = \begin{cases} X_{new} & F(X_{new}) < F(X_{old}) \\ X_{old} & \text{others} \end{cases} \quad (15)$$

where  $X_{new}$  represents the new search agent generated using the orthogonal policy and  $X_{old}$  is the search agent without the orthogonal policy.  $F$  represents the fitness function.

---

**Algorithm 2** Pseudo-code for OLCPA

---

Initialize population size  $Num$ , the problem dimension  $dim$ , and the maximum number of evaluations  $MaxFes$

**While** ( $t \leq MaxFes$ )

**For**  $i = 1: Num$

    Calculation of individual fitness values

    Update  $X_{best}$

**End for**

**For**  $j = 1: dim$

      Update  $X_1, X_2$

      Calculate  $X_j^i$  by Equation (1)

**End for**

**For**  $i = 1: Num$

    Update  $S$

**If**  $|S| < \frac{2}{3}a$

      Calculate the current agent's position by Equation (8)

**End if**

**If**  $|S| \geq \frac{2}{3}a$

        Calculate the current agent's position by Equation (13)

**End if**

**End for**

**Execute an orthogonal strategy**

**Update the current search agent**

$t = t + 1$

**End while**

Return  $X_{best}$

---

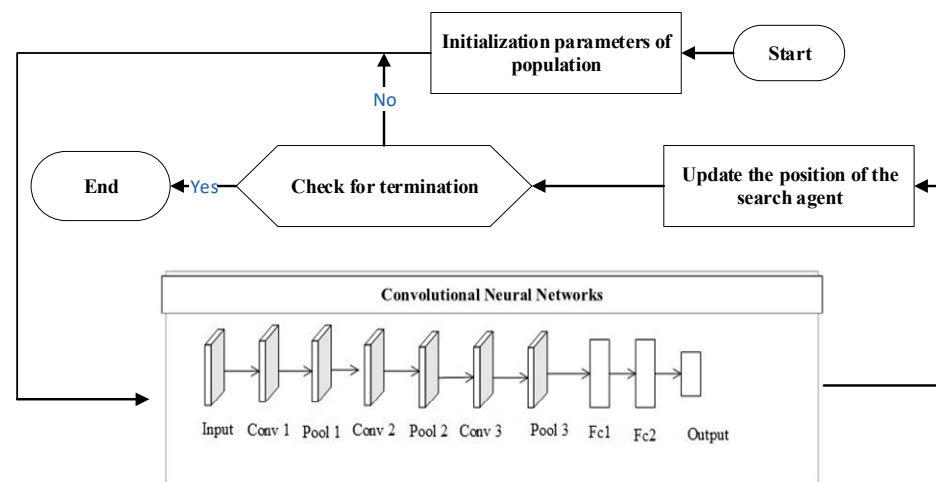
#### 4. The Design of the OLCPA-CNN Model

This section describes the OLCPA formation process, and the classification error rate of CNN is used as the fitness function. Then, the parameters of the CNN are optimized according to the optimal solution generated by OLCPA to obtain the OLCPA-CNN model and evaluate this model.

##### 4.1. The Network Structure of CNN

The nine-layer CNN model used in this paper is displayed in Figure 1, and the overall structure consists of three sets of convolutional pooling layers and two fully connected layers, where each set of convolutional pooling layers consists of one convolutional layer and one pooling layer. Data are transmitted to the input layer, which undergoes convolution and pooling operations to achieve the mapping of different functions and simultaneously extract useful features, and then achieves the classification purpose in the fully connected layer. Figure 3 depicts the process of OLCPA-CNN being designed. The search agent of OLCPA represents the parameters of the CNN, and the optimized parameters of the CNN are obtained through iterative updates of the position of the search agent of OLCPA.





**Figure 3.** The formation process of the OLCPA-CNN model.

4.2. Hyperparameter Optimization

Parameter tuning occupies a crucial place in deep learning classification [64,65]. OLCPA optimizes the parameters of the CNN structure by iteratively updating the solution. In this paper, the parameters to be optimized are concentrated in the following layers: a three-layer convolutional layer and a two-layer fully connected layer. The optimization of hyperparameters in the CNN includes the number and size of convolutional kernels for each convolutional layer, the number of first fully connected layers, the learning rate, the number of epochs, and L2 regularization. The optimized parameters and their ranges are described in Table 1. The parameters of OLCPA are as follows: the number of populations is 5, the maximum number of iterations is 20, and the dimensionality is set to 10. Because the number of parameters to be optimized is closely related to the parameters of the CNN, its bounded range is related to the range of parameter variations.

**Table 1.** Hyperparameter range of CNN structure.

Architecture	Hyperparameter	Range
CNN	Number of convolution kernels	[1, 15]
	Size of the convolution kernels	[1, 128]
	Number of nodes of the first fully connected layer	[0, 5000]
	Number of epochs	[1, 40]
	Learning rate	[0.0001, 0.01]
	L2 regularization	[0.001, 0.01]

4.3. Fitness Function

The fitness function is essentially an evaluation function that evaluates the goodness of an algorithmic solution by measuring the fitness of individuals in the population [66]. In this article, the fitness function is the classification error rate, closely related to classification accuracy (ACC). The correct classification rate is calculated based on the confusion matrix. Table 2 shows the expressions of the confusion matrix, where TP means that both the actual and predicted values are positive, TN means that both values are negative, FN means that the predicted value is negative and the real value is positive, and FP means that the predicted value is positive and the real value is negative. The formula for ACC is shown in Equation (16), and the corresponding fitness function is calculated as shown in Equation (17). The relationship between the value of the fitness function and the classification error rate is proportional; when the value of the fitness function becomes smaller,

the classification error rate also becomes smaller, which reflects that the quality of the individual solution is good and it is beneficial to the optimization of CNN parameters.

$$\text{ACC} = \frac{TP + TN}{TP + FP + FN + TN} \quad (16)$$

$$\text{Fitness} = 1 - \text{ACC} \quad (17)$$

**Table 2.** Confusion matrix.

	Actual Positive	Actual Negative
Predicted positive	$TP$	$FP$
Predicted negative	$FN$	$TN$

## 5. Experimental Design and Results of OLCPA

In this section, we compare the OLCPA algorithm with other algorithms on the IEEE CEC2017 benchmark functions to verify the performance of OLCPA. These algorithms were tested using MATLAB R2018b with 128GB of RAM and an Intel(R) Xeon(R) Silver 4110 CPU on Windows Server 2016. The assessment of AI-based approaches through fair procedures can advance replicability, transparency, research standards, and public confidence [67–69]. Comparing computational methods using the same criteria allows us to establish unbiased assessments [70,71]. We conducted our trials in a manner consistent with fair comparison principles. The parameters involved in the experiments are as follows: the number of populations  $N$ , dimension  $D$ , the maximum number of evaluations  $\text{MaxFes}$ , and the number of independent runs of the algorithm  $\text{Num}$ . Table 3 shows the values of these parameters.

**Table 3.** Parameters of an algorithm comparison experiment.

<b>N</b>	<b>D</b>	<b>MaxFes</b>	<b>Num</b>
30	30	300,000	30

### 5.1. Benchmark Function

In this subsection, the IEEE CEC2017 benchmark functions [72] are used to test the performance of the OLCPA. These functions are classified into the following categories: unimodal functions (F1–F3), multimodal functions (F4–F10), hybrid functions (F11–F20), and complex functions (F21–F30).

### 5.2. Scalability Test

This section tests OLCPA and CPA in 50 and 100 dimensions under the same experimental conditions. The dimensionality tests for scalability are mainly to verify the performance of OLCPA when coping with different dimensions. Table 4 shows the comparison results of these two algorithms in different dimensions. Avg and Std denote the mean and standard deviation of the experimental results, respectively. The experimental results reveal that OLCPA can also perform well in high dimensionality. The experimental results in the table show that the quality of most solutions of OLCPA is significantly stronger than that of CPA in 50 and 100 dimensions, which indicates that the improved OLCPA algorithm based on CPA is effective and the performance of OLCPA is stronger.

**Table 4.** Comparison of OLCPA and CPA dimensions.

F	Algorithm	Dim = 50		Dim = 100	
		Avg	Std	Avg	Std
F1	CPA	$3.918 \times 10^3$	$5.613 \times 10^3$	$1.0134 \times 10^4$	$1.3351 \times 10^4$
	OLCPA	$2.045 \times 10^3$	$2.522 \times 10^3$	$7.4512 \times 10^3$	$6.3808 \times 10^3$
F2	CPA	$3.212 \times 10^3$	$9.123 \times 10^3$	$1.2924 \times 10^{13}$	$6.9606 \times 10^{13}$
	OLCPA	$2.804 \times 10^2$	$1.873 \times 10^2$	$2.7373 \times 10^{10}$	$1.0467 \times 10^{11}$
F3	CPA	$3.000 \times 10^2$	$1.723 \times 10^{-6}$	$2.3131 \times 10^3$	$1.0879 \times 10^3$
	OLCPA	$3.000 \times 10^2$	$1.425 \times 10^{-9}$	$3.0000 \times 10^2$	$2.1451 \times 10^{-8}$
F4	CPA	$5.069 \times 10^2$	$5.491 \times 10$	$6.2239 \times 10^2$	$3.7274 \times 10$
	OLCPA	$4.769 \times 10^2$	$3.367 \times 10$	$6.3116 \times 10^2$	$5.2452 \times 10$
F5	CPA	$7.157 \times 10^2$	$2.635 \times 10$	$1.0656 \times 10^3$	$7.0536 \times 10$
	OLCPA	$7.393 \times 10^2$	$3.785 \times 10$	$1.1193 \times 10^3$	$5.8359 \times 10$
F6	CPA	$6.000 \times 10^2$	$2.072 \times 10^{-4}$	$6.0000 \times 10^2$	$3.5520 \times 10^{-3}$
	OLCPA	$6.000 \times 10^2$	$1.946 \times 10^{-13}$	$6.0000 \times 10^2$	$2.6953 \times 10^{-13}$
F7	CPA	$9.926 \times 10^2$	$4.689 \times 10$	$1.4700 \times 10^3$	$9.1916 \times 10$
	OLCPA	$1.012 \times 10^3$	$4.161 \times 10$	$1.5541 \times 10^3$	$1.2035 \times 10^2$
F8	CPA	$1.017 \times 10^3$	$3.768 \times 10$	$1.3892 \times 10^3$	$7.3342 \times 10$
	OLCPA	$1.018 \times 10^3$	$3.874 \times 10$	$1.4321 \times 10^3$	$6.7090 \times 10$
F9	CPA	$6.632 \times 10^3$	$1.662 \times 10^3$	$1.6982 \times 10^4$	$1.9060 \times 10^3$
	OLCPA	$6.566 \times 10^3$	$1.940 \times 10^3$	$1.6940 \times 10^4$	$1.8023 \times 10^3$
F10	CPA	$5.857 \times 10^3$	$6.643 \times 10^2$	$1.2755 \times 10^4$	$1.0274 \times 10^3$
	OLCPA	$5.598 \times 10^3$	$7.011 \times 10^2$	$1.2856 \times 10^4$	$1.2503 \times 10^3$
F11	CPA	$1.226 \times 10^3$	$3.445 \times 10$	$1.5628 \times 10^3$	$1.0329 \times 10^2$
	OLCPA	$1.228 \times 10^3$	$2.865 \times 10$	$1.5100 \times 10^3$	$1.1924 \times 10^2$
F12	CPA	$3.216 \times 10^6$	$2.379 \times 10^6$	$8.5478 \times 10^6$	$3.7663 \times 10^6$
	OLCPA	$2.110 \times 10^6$	$1.443 \times 10^6$	$4.4447 \times 10^6$	$2.1365 \times 10^6$
F13	CPA	$8.034 \times 10^3$	$8.404 \times 10^3$	$6.9597 \times 10^3$	$5.4080 \times 10^3$
	OLCAP	$5.223 \times 10^3$	$3.526 \times 10^3$	$4.9575 \times 10^3$	$3.4951 \times 10^3$
F14	CPA	$4.017 \times 10^4$	$2.045 \times 10^4$	$1.0750 \times 10^5$	$3.2124 \times 10^4$
	OLCAP	$8.857 \times 10^3$	$5.974 \times 10^3$	$3.4510 \times 10^4$	$6.5897 \times 10^3$
F15	CPA	$8.161 \times 10^3$	$5.507 \times 10^3$	$4.0322 \times 10^3$	$3.0081 \times 10^3$
	OLCPA	$9.451 \times 10^3$	$5.953 \times 10^3$	$2.7858 \times 10^3$	$1.2726 \times 10^3$
F16	CPA	$3.647 \times 10^3$	$4.486 \times 10^2$	$6.1075 \times 10^3$	$6.1143 \times 10^2$
	OLCPA	$3.453 \times 10^3$	$3.315 \times 10^2$	$5.9226 \times 10^3$	$5.8144 \times 10^2$
F17	CPA	$3.034 \times 10^3$	$3.329 \times 10^2$	$4.9720 \times 10^3$	$5.3734 \times 10^2$
	OLCPA	$3.109 \times 10^3$	$3.106 \times 10^2$	$4.8910 \times 10^3$	$6.0655 \times 10^2$
F18	CPA	$1.320 \times 10^5$	$2.658 \times 10^4$	$2.5015 \times 10^5$	$9.0799 \times 10^4$
	OLCPA	$4.246 \times 10^4$	$1.368 \times 10^4$	$1.3636 \times 10^5$	$2.4613 \times 10^4$
F19	CPA	$2.097 \times 10^4$	$9.306 \times 10^3$	$5.9069 \times 10^3$	$4.3932 \times 10^3$
	OLCPA	$2.660 \times 10^4$	$8.118 \times 10^3$	$3.8745 \times 10^3$	$1.7514 \times 10^3$
F20	CPA	$3.055 \times 10^3$	$3.027 \times 10^2$	$5.1363 \times 10^3$	$3.9298 \times 10^2$
	OLCPA	$2.891 \times 10^3$	$2.578 \times 10^2$	$5.1443 \times 10^3$	$4.6463 \times 10^2$
F21	CPA	$2.515 \times 10^3$	$3.993 \times 10$	$2.8858 \times 10^3$	$8.8322 \times 10$
	OLCPA	$2.527 \times 10^3$	$4.671 \times 10$	$2.8781 \times 10^3$	$7.3269 \times 10$
F22	CPA	$7.882 \times 10^3$	$1.718 \times 10^3$	$1.6525 \times 10^4$	$1.2773 \times 10^3$
	OLCPA	$7.908 \times 10^3$	$1.320 \times 10^3$	$1.6529 \times 10^4$	$9.8827 \times 10^2$
F23	CPA	$2.979 \times 10^3$	$4.502 \times 10$	$3.1604 \times 10^3$	$6.6818 \times 10$
	OLCPA	$3.008 \times 10^3$	$4.940 \times 10$	$3.1599 \times 10^3$	$7.3210 \times 10$
F24	CPA	$3.498 \times 10^3$	$1.719 \times 10^2$	$3.8236 \times 10^3$	$8.3982 \times 10$
	OLCPA	$3.566 \times 10^3$	$1.472 \times 10^2$	$3.8711 \times 10^3$	$8.9281 \times 10$
F25	CPA	$3.048 \times 10^3$	$4.905 \times 10$	$3.2936 \times 10^3$	$7.0202 \times 10$
	OLCPA	$3.052 \times 10^3$	$3.882 \times 10$	$3.2933 \times 10^3$	$7.0629 \times 10$
F26	CPA	$4.078 \times 10^3$	$2.100 \times 10^3$	$1.2307 \times 10^4$	$3.3495 \times 10^3$
	OLCPA	$5.254 \times 10^3$	$2.960 \times 10^3$	$1.3663 \times 10^4$	$2.7772 \times 10^3$

Table 4. Cont.

F	Algorithm	Dim = 50		Dim = 100	
		Avg	Std	Avg	Std
F27	CPA	$3.519 \times 10^3$	$1.029 \times 10^2$	$3.5698 \times 10^3$	$8.1696 \times 10$
	OLCPA	$3.532 \times 10^3$	$9.428 \times 10$	$3.6442 \times 10^3$	$8.7973 \times 10$
F28	CPA	$3.296 \times 10^3$	$2.671 \times 10$	$3.3820 \times 10^3$	$3.4844 \times 10$
	OLCPA	$3.290 \times 10^3$	$2.094 \times 10$	$3.3647 \times 10^3$	$4.4484 \times 10$
F29	CPA	$4.233 \times 10^3$	$2.991 \times 10^2$	$6.7623 \times 10^3$	$4.7534 \times 10^2$
	OLCPA	$4.072 \times 10^3$	$2.955 \times 10^2$	$6.9277 \times 10^3$	$5.1706 \times 10^2$
F30	CPA	$9.734 \times 10^5$	$2.383 \times 10^5$	$1.4286 \times 10^4$	$4.3046 \times 10^3$
	OLCPA	$8.788 \times 10^5$	$1.518 \times 10^5$	$1.3530 \times 10^4$	$4.6178 \times 10^3$

5.3. Comparison with Conventional and Advanced Algorithms

In this section, to test the performance of OLCPA, it is compared with 11 algorithms: CCMWOA [73], IGWO [74], CCMSCSA [75], BMWOA [76], CMFO [77], CESCA [78], GCHHO [79], DE [80], MFO [81], HGS [30], and CPA [33]. To ensure the reliability of the experiment, this experiment was conducted under the same experimental conditions. The results of comparing OLCPA with the algorithms mentioned above are listed in Table 5. The last three columns of the table are Rank, the symbol “+/-/-”, and Avg, where Rank represents the Friedman test, “+/-/-” represents the number of functions in which OLCPA is stronger than, equal to, or not stronger than the other algorithms for the 30 benchmark functions, and Avg represents the average of the benchmark function test results.

Table 5. Comparison of OLCPA with other algorithms.

Algorithm	F1		F2		F3	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$2.6417 \times 10^3$	$2.6210 \times 10^3$	$2.0000 \times 10^2$	$6.1889 \times 10^{-6}$	$3.0000 \times 10^2$	$3.3039 \times 10^{-10}$
CCMWOA	$2.0529 \times 10^{10}$	$4.7896 \times 10^9$	$3.9233 \times 10^{38}$	$1.8806 \times 10^{39}$	$7.7098 \times 10^4$	$6.7169 \times 10^3$
IGWO	$1.6989 \times 10^6$	$8.5027 \times 10^5$	$2.0146 \times 10^{13}$	$8.4520 \times 10^{13}$	$1.4554 \times 10^3$	$6.7162 \times 10^2$
CCMSCSA	$3.1354 \times 10^3$	$3.1637 \times 10^3$	$7.5324 \times 10^{10}$	$2.3962 \times 10^{11}$	$3.4410 \times 10^2$	$3.2511 \times 10^1$
BMWOA	$2.0649 \times 10^8$	$8.6683 \times 10^7$	$5.7378 \times 10^{22}$	$2.6681 \times 10^{23}$	$7.0802 \times 10^4$	$9.5290 \times 10^3$
CMFO	$2.0503 \times 10^8$	$4.7086 \times 10^8$	$3.8481 \times 10^{37}$	$2.0987 \times 10^{38}$	$1.1542 \times 10^5$	$4.6442 \times 10^4$
CESCA	$5.7624 \times 10^{10}$	$4.6938 \times 10^9$	$5.0711 \times 10^{45}$	$1.0293 \times 10^{46}$	$1.0551 \times 10^5$	$1.4524 \times 10^4$
GCHHO	$4.6746 \times 10^3$	$5.9329 \times 10^3$	$4.0569 \times 10^5$	$9.1569 \times 10^5$	$5.5167 \times 10^2$	$1.6374 \times 10^2$
DE	$2.2872 \times 10^3$	$3.7753 \times 10^3$	$1.3602 \times 10^{21}$	$3.6101 \times 10^{21}$	$1.9826 \times 10^4$	$4.4953 \times 10^3$
MFO	$1.3517 \times 10^{10}$	$8.8111 \times 10^9$	$1.1274 \times 10^{39}$	$6.1676 \times 10^{39}$	$1.1049 \times 10^5$	$8.7936 \times 10^4$
HGS	$1.3861 \times 10^7$	$7.5867 \times 10^7$	$1.3521 \times 10^{16}$	$5.1457 \times 10^{16}$	$2.6774 \times 10^3$	$5.6921 \times 10^3$
CPA	$5.3939 \times 10^3$	$5.9328 \times 10^3$	$2.0098 \times 10^2$	$3.7267 \times 10^0$	$3.0000 \times 10^2$	$1.4672 \times 10^{-7}$
	F4		F5		F6	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$4.4457 \times 10^2$	$3.6133 \times 10^1$	$6.2701 \times 10^2$	$2.3849 \times 10^1$	$6.0000 \times 10^2$	$3.5452 \times 10^{-13}$
CCMWOA	$3.7003 \times 10^3$	$1.4526 \times 10^3$	$8.3497 \times 10^2$	$3.3283 \times 10^1$	$6.7164 \times 10^2$	$7.8115 \times 10^0$
IGWO	$5.0643 \times 10^2$	$2.3656 \times 10^1$	$6.1178 \times 10^2$	$1.6784 \times 10^1$	$6.2273 \times 10^2$	$5.5501 \times 10^0$
CCMSCSA	$4.9943 \times 10^2$	$2.7486 \times 10^1$	$5.8212 \times 10^2$	$2.3957 \times 10^1$	$6.0043 \times 10^2$	$2.9648 \times 10^{-1}$
BMWOA	$6.0019 \times 10^2$	$3.8438 \times 10^1$	$7.7892 \times 10^2$	$5.5275 \times 10^1$	$6.6611 \times 10^2$	$1.1204 \times 10^1$
CMFO	$5.6429 \times 10^2$	$6.6364 \times 10^1$	$7.2496 \times 10^2$	$5.0447 \times 10^1$	$6.5202 \times 10^2$	$9.3009 \times 10^0$
CESCA	$1.5015 \times 10^4$	$2.3052 \times 10^3$	$9.6832 \times 10^2$	$2.4033 \times 10^1$	$7.0496 \times 10^2$	$5.2640 \times 10^0$
GCHHO	$4.9548 \times 10^2$	$2.8019 \times 10^1$	$7.1025 \times 10^2$	$4.2271 \times 10^1$	$6.5178 \times 10^2$	$6.9046 \times 10^0$
DE	$4.9088 \times 10^2$	$9.5252 \times 10^0$	$6.0806 \times 10^2$	$9.1168 \times 10^0$	$6.0000 \times 10^2$	$0.0000 \times 10^0$
MFO	$1.3689 \times 10^3$	$8.5540 \times 10^2$	$7.0259 \times 10^2$	$6.0823 \times 10^1$	$6.4206 \times 10^2$	$1.1753 \times 10^1$
HGS	$4.7827 \times 10^2$	$2.7144 \times 10^1$	$6.3080 \times 10^2$	$2.8629 \times 10^1$	$6.0152 \times 10^2$	$1.9161 \times 10^0$
CPA	$4.8346 \times 10^2$	$2.5598 \times 10^1$	$6.2974 \times 10^2$	$2.6272 \times 10^1$	$6.0000 \times 10^2$	$1.0003 \times 10^{-7}$

Table 5. Cont.

	F7		F8		F9	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$8.5580 \times 10^2$	$3.3574 \times 10^1$	$9.0241 \times 10^2$	$1.6026 \times 10^1$	$2.6955 \times 10^3$	$5.2050 \times 10^2$
CCMWOA	$1.2785 \times 10^3$	$7.1917 \times 10^1$	$1.0445 \times 10^3$	$2.5360 \times 10^1$	$7.7640 \times 10^3$	$1.4347 \times 10^3$
IGWO	$9.0405 \times 10^2$	$5.2744 \times 10^1$	$8.9353 \times 10^2$	$2.1787 \times 10^1$	$2.8209 \times 10^3$	$8.7020 \times 10^2$
CCMSCSA	$8.0574 \times 10^2$	$1.7076 \times 10^1$	$9.0305 \times 10^2$	$2.9632 \times 10^1$	$9.8573 \times 10^2$	$7.2563 \times 10^1$
BMWOA	$1.1733 \times 10^3$	$1.0644 \times 10^2$	$1.0081 \times 10^3$	$3.0517 \times 10^1$	$7.2354 \times 10^3$	$8.8766 \times 10^2$
CMFO	$1.2808 \times 10^3$	$1.5349 \times 10^2$	$9.5931 \times 10^2$	$3.8407 \times 10^1$	$4.7987 \times 10^3$	$1.1702 \times 10^3$
CESCA	$1.5498 \times 10^3$	$5.0905 \times 10^1$	$1.1778 \times 10^3$	$1.9507 \times 10^1$	$1.5424 \times 10^4$	$1.2474 \times 10^3$
GCHHO	$1.0821 \times 10^3$	$1.0330 \times 10^2$	$9.4369 \times 10^2$	$2.0730 \times 10^1$	$4.7578 \times 10^3$	$5.8635 \times 10^2$
DE	$8.4129 \times 10^2$	$1.0450 \times 10^1$	$9.0777 \times 10^2$	$8.5623 \times 10^0$	$9.0000 \times 10^2$	$1.0765 \times 10^{-13}$
MFO	$1.1498 \times 10^3$	$2.0356 \times 10^2$	$1.0177 \times 10^3$	$4.6613 \times 10^1$	$7.1329 \times 10^3$	$1.7975 \times 10^3$
HGS	$8.9314 \times 10^2$	$5.1096 \times 10^1$	$9.0545 \times 10^2$	$2.2953 \times 10^1$	$3.5491 \times 10^3$	$8.3458 \times 10^2$
CPA	$8.4269 \times 10^2$	$2.7130 \times 10^1$	$9.0484 \times 10^2$	$2.1313 \times 10^1$	$2.3193 \times 10^3$	$6.1060 \times 10^2$
	F10		F11		F12	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$3.7605 \times 10^3$	$3.8850 \times 10^2$	$1.1756 \times 10^3$	$3.4123 \times 10^1$	$6.5992 \times 10^5$	$4.6976 \times 10^5$
CCMWOA	$7.0372 \times 10^3$	$6.1016 \times 10^2$	$3.1558 \times 10^3$	$5.8702 \times 10^2$	$2.0126 \times 10^9$	$1.4565 \times 10^9$
IGWO	$4.4687 \times 10^3$	$5.9061 \times 10^2$	$1.2642 \times 10^3$	$2.8641 \times 10^1$	$1.5414 \times 10^7$	$1.5084 \times 10^7$
CCMSCSA	$4.6758 \times 10^3$	$6.1466 \times 10^2$	$1.1870 \times 10^3$	$3.1743 \times 10^1$	$1.1060 \times 10^6$	$8.7007 \times 10^5$
BMWOA	$7.4949 \times 10^3$	$5.9325 \times 10^2$	$1.6517 \times 10^3$	$1.6384 \times 10^2$	$7.8078 \times 10^7$	$5.9556 \times 10^7$
CMFO	$7.3777 \times 10^3$	$1.2921 \times 10^3$	$4.6678 \times 10^3$	$3.4534 \times 10^3$	$4.0157 \times 10^7$	$1.2585 \times 10^8$
CESCA	$8.7430 \times 10^3$	$2.2735 \times 10$	$1.0664 \times 10^4$	$1.6523 \times 10^3$	$1.5622 \times 10^0$	$1.5369 \times 10^9$
GCHHO	$5.1344 \times 10^3$	$6.1384 \times 10^2$	$1.2339 \times 10^3$	$5.1150 \times 10^1$	$9.6394 \times 10^5$	$7.5930 \times 10^5$
DE	$5.9154 \times 10^3$	$3.1146 \times 10^2$	$1.1611 \times 10^3$	$2.2327 \times 10^1$	$1.6551 \times 10^6$	$8.2025 \times 10^5$
MFO	$5.6084 \times 10^3$	$8.5316 \times 10^2$	$4.6351 \times 10^3$	$4.6023 \times 10^3$	$1.9357 \times 10^8$	$3.4217 \times 10^8$
HGS	$3.9194 \times 10^3$	$4.7298 \times 10^2$	$1.2032 \times 10^3$	$3.5853 \times 10^1$	$7.1069 \times 10^5$	$5.9578 \times 10^5$
CPA	$3.6850 \times 10^3$	$4.6305 \times 10^2$	$1.1700 \times 10^3$	$3.4461 \times 10^1$	$1.6148 \times 10^6$	$1.2540 \times 10^6$
	F13		F14		F15	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$4.4219 \times 10^3$	$2.9844 \times 10^3$	$2.9128 \times 10^3$	$1.0472 \times 10^3$	$3.2767 \times 10^3$	$2.4501 \times 10^3$
CCMWOA	$1.5038 \times 10^8$	$2.1857 \times 10^8$	$1.2576 \times 10^6$	$8.9707 \times 10^5$	$5.8975 \times 10^6$	$9.3662 \times 10^6$
IGWO	$2.8168 \times 10^5$	$3.9417 \times 10^5$	$5.3978 \times 10^4$	$3.4567 \times 10^4$	$5.6870 \times 10^4$	$2.9361 \times 10^4$
CCMSCSA	$1.3367 \times 10^4$	$1.1171 \times 10^4$	$1.6896 \times 10^4$	$1.5715 \times 10^4$	$3.0463 \times 10^3$	$2.0005 \times 10^3$
BMWOA	$4.3710 \times 10^5$	$7.0418 \times 10^5$	$9.4482 \times 10^5$	$8.0898 \times 10^5$	$1.7030 \times 10^5$	$2.7468 \times 10^5$
CMFO	$3.7917 \times 10^7$	$1.9343 \times 10^8$	$3.5943 \times 10^5$	$8.2977 \times 10^5$	$3.0292 \times 10^4$	$3.2825 \times 10^4$
CESCA	$1.3433 \times 10^{10}$	$3.9068 \times 10^9$	$6.5888 \times 10^6$	$2.8850 \times 10^6$	$4.5428 \times 10^8$	$1.8134 \times 10^8$
GCHHO	$1.2843 \times 10^4$	$1.5165 \times 10^4$	$3.4759 \times 10^4$	$2.5482 \times 10^4$	$6.3001 \times 10^3$	$6.5459 \times 10^3$
DE	$2.9103 \times 10^4$	$1.6893 \times 10^4$	$4.9826 \times 10^4$	$2.5793 \times 10^4$	$8.6203 \times 10^3$	$5.3792 \times 10^3$
MFO	$3.5810 \times 10^6$	$1.3021 \times 10^7$	$2.3452 \times 10^5$	$6.2121 \times 10^5$	$6.7501 \times 10^4$	$6.6285 \times 10^4$
HGS	$2.6168 \times 10^4$	$2.4091 \times 10^4$	$5.3803 \times 10^4$	$4.0736 \times 10^4$	$1.7192 \times 10^4$	$1.5459 \times 10^4$
CPA	$5.8096 \times 10^3$	$1.1246 \times 10^4$	$7.0490 \times 10^3$	$4.9848 \times 10^3$	$2.2795 \times 10^3$	$9.4388 \times 10^2$

Table 5. Cont.

	F16		F17		F18	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$2.5673 \times 10^3$	$3.5254 \times 10^2$	$2.0234 \times 10^3$	$1.5350 \times 10^2$	$3.3044 \times 10^4$	$1.4827 \times 10^4$
CCMWOA	$3.9526 \times 10^3$	$6.7820 \times 10^2$	$2.7756 \times 10^3$	$3.8592 \times 10^2$	$1.0532 \times 10^7$	$1.0505 \times 10^7$
IGWO	$2.5650 \times 10^3$	$3.5963 \times 10^2$	$2.0210 \times 10^3$	$1.4303 \times 10^2$	$4.9309 \times 10^5$	$4.1194 \times 10^5$
CCMSCSA	$2.5013 \times 10^3$	$2.6859 \times 10^2$	$2.0794 \times 10^3$	$1.8933 \times 10^2$	$1.6964 \times 10^5$	$1.5010 \times 10^5$
BMWOA	$3.4890 \times 10^3$	$5.2235 \times 10^2$	$2.4671 \times 10^3$	$2.1627 \times 10^2$	$3.2300 \times 10^6$	$3.6494 \times 10^6$
CMFO	$2.9376 \times 10^3$	$5.1792 \times 10^2$	$2.4441 \times 10^3$	$3.0889 \times 10^2$	$2.8221 \times 10^6$	$5.1454 \times 10^6$
CESCA	$5.9581 \times 10^3$	$4.8739 \times 10^2$	$4.4216 \times 10^3$	$4.3712 \times 10^2$	$5.7643 \times 10^7$	$2.6176 \times 10^7$
GCHHO	$2.7374 \times 10^3$	$2.7503 \times 10^2$	$2.3088 \times 10^3$	$2.6833 \times 10^2$	$2.5047 \times 10^5$	$3.0345 \times 10^5$
DE	$2.0652 \times 10^3$	$1.4220 \times 10^2$	$1.8272 \times 10^3$	$4.6232 \times 10^1$	$3.2091 \times 10^5$	$1.8003 \times 10^5$
MFO	$3.1336 \times 10^3$	$4.4336 \times 10^2$	$2.5667 \times 10^3$	$3.1189 \times 10^2$	$1.6242 \times 10^6$	$3.0380 \times 10^6$
HGS	$2.6782 \times 10^3$	$3.3225 \times 10^2$	$2.2166 \times 10^3$	$2.5020 \times 10^2$	$2.8938 \times 10^5$	$2.7168 \times 10^5$
CPA	$2.7639 \times 10^3$	$2.9671 \times 10^2$	$2.1603 \times 10^3$	$2.6412 \times 10^2$	$1.0822 \times 10^5$	$6.6646 \times 10^4$
	F19		F20		F21	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$4.2596 \times 10^3$	$2.0678 \times 10^3$	$2.3366 \times 10^3$	$1.1975 \times 10^2$	$2.4210 \times 10^3$	$2.6344 \times 10^1$
CCMWOA	$5.5422 \times 10^6$	$9.1972 \times 10^6$	$2.7663 \times 10^3$	$1.8365 \times 10^2$	$2.6152 \times 10^3$	$6.4209 \times 10^1$
IGWO	$2.2651 \times 10^5$	$2.5430 \times 10^5$	$2.3539 \times 10^3$	$1.2977 \times 10^2$	$2.3977 \times 10^3$	$2.4054 \times 10^1$
CCMSCSA	$6.5091 \times 10^3$	$5.1531 \times 10^3$	$2.3399 \times 10^3$	$1.2859 \times 10^2$	$2.3752 \times 10^3$	$1.8620 \times 10^1$
BMWOA	$8.1030 \times 10^5$	$1.1393 \times 10^6$	$2.7627 \times 10^3$	$1.8733 \times 10^2$	$2.5221 \times 10^3$	$5.0213 \times 10^1$
CMFO	$4.4672 \times 10^4$	$7.9129 \times 10^4$	$2.7796 \times 10^3$	$1.8148 \times 10^2$	$2.4958 \times 10^3$	$3.7613 \times 10^1$
CESCA	$1.3527 \times 10^9$	$4.4096 \times 10^8$	$3.1735 \times 10^3$	$1.3671 \times 10^2$	$2.7653 \times 10^3$	$3.4531 \times 10^1$
GCHHO	$6.1960 \times 10^3$	$5.0850 \times 10^3$	$2.5673 \times 10^3$	$1.8589 \times 10^2$	$2.4895 \times 10^3$	$5.0529 \times 10^1$
DE	$8.0940 \times 10^3$	$5.1894 \times 10^3$	$2.1309 \times 10^3$	$8.2781 \times 10^1$	$2.4037 \times 10^3$	$9.0200 \times 10^0$
MFO	$1.1628 \times 10^7$	$3.7701 \times 10^7$	$2.7001 \times 10^3$	$2.2561 \times 10^2$	$2.5056 \times 10^3$	$4.5377 \times 10^1$
HGS	$1.2762 \times 10^4$	$1.5843 \times 10^4$	$2.4769 \times 10^3$	$1.7556 \times 10^2$	$2.4252 \times 10^3$	$2.8533 \times 10^1$
CPA	$5.3098 \times 10^3$	$1.9655 \times 10^3$	$2.4748 \times 10^3$	$1.4970 \times 10^2$	$2.4060 \times 10^3$	$7.0170 \times 10^1$
	F22		F23		F24	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$3.9509 \times 10^3$	$1.9526 \times 10^3$	$2.7595 \times 10^3$	$3.2074 \times 10^1$	$3.1311 \times 10^3$	$9.8525 \times 10^1$
CCMWOA	$7.3798 \times 10^3$	$1.3564 \times 10^3$	$3.1950 \times 10^3$	$1.1166 \times 10^2$	$3.3448 \times 10^3$	$1.1703 \times 10^2$
IGWO	$2.3179 \times 10^3$	$3.7459 \times 10^1$	$2.7712 \times 10^3$	$3.0804 \times 10^1$	$2.9433 \times 10^3$	$3.3180 \times 10^1$
CCMSCSA	$2.3011 \times 10^3$	$1.8012 \times 10^0$	$2.7389 \times 10^3$	$2.2920 \times 10^1$	$2.9120 \times 10^3$	$2.9458 \times 10^1$
BMWOA	$6.0884 \times 10^3$	$3.1127 \times 10^3$	$2.9482 \times 10^3$	$7.9106 \times 10^1$	$3.1150 \times 10^3$	$7.4716 \times 10^1$
CMFO	$5.4333 \times 10^3$	$2.9116 \times 10^3$	$2.9734 \times 10^3$	$7.2165 \times 10^1$	$3.1313 \times 10^3$	$1.1677 \times 10^2$
CESCA	$9.5457 \times 10^3$	$5.7616 \times 10^2$	$3.4764 \times 10^3$	$4.8311 \times 10^1$	$3.4817 \times 10^3$	$3.3877 \times 10^1$
GCHHO	$4.1361 \times 10^3$	$2.1478 \times 10^3$	$2.9327 \times 10^3$	$6.8914 \times 10^1$	$3.0983 \times 10^3$	$7.3773 \times 10^1$
DE	$3.7200 \times 10^3$	$1.7703 \times 10^3$	$2.7561 \times 10^3$	$8.0338 \times 10^0$	$2.9580 \times 10^3$	$1.1083 \times 10^1$
MFO	$6.4721 \times 10^3$	$1.7866 \times 10^3$	$2.8414 \times 10^3$	$3.5967 \times 10^1$	$2.9872 \times 10^3$	$3.6604 \times 10^1$
HGS	$4.6540 \times 10^3$	$1.5057 \times 10^3$	$2.7673 \times 10^3$	$2.8088 \times 10^1$	$3.0210 \times 10^3$	$4.9364 \times 10^1$
CPA	$3.1890 \times 10^3$	$1.6475 \times 10^3$	$2.7663 \times 10^3$	$3.4140 \times 10^1$	$3.0664 \times 10^3$	$6.5262 \times 10^1$

Table 5. Cont.

	F25		F26		F27	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$2.8894 \times 10^3$	$7.7206 \times 10^0$	$4.4389 \times 10^3$	$1.1528 \times 10^3$	$3.2423 \times 10^3$	$1.9934 \times 10^1$
CCMWOA	$3.3774 \times 10^3$	$1.1165 \times 10^2$	$8.7361 \times 10^3$	$9.8681 \times 10^2$	$3.5982 \times 10^3$	$1.5901 \times 10^2$
IGWO	$2.9064 \times 10^3$	$1.6945 \times 10^1$	$4.8594 \times 10^3$	$2.8776 \times 10^2$	$3.2367 \times 10^3$	$1.3129 \times 10^1$
CCMSCSA	$2.9035 \times 10^3$	$1.6558 \times 10^1$	$3.5575 \times 10^3$	$1.1866 \times 10^3$	$3.2607 \times 10^3$	$2.5012 \times 10^1$
BMWOA	$3.0206 \times 10^3$	$3.3567 \times 10^1$	$6.8057 \times 10^3$	$1.2593 \times 10^3$	$3.3084 \times 10^3$	$5.7831 \times 10^1$
CMFO	$2.9541 \times 10^3$	$3.6301 \times 10^1$	$6.8104 \times 10^3$	$7.5311 \times 10^2$	$3.4320 \times 10^3$	$1.5391 \times 10^2$
CESCA	$5.5207 \times 10^3$	$4.9939 \times 10^2$	$1.1158 \times 10^4$	$5.9604 \times 10^2$	$3.6926 \times 10^3$	$7.6258 \times 10^1$
GCHHO	$2.8956 \times 10^3$	$1.6050 \times 10^1$	$6.0549 \times 10^3$	$1.2718 \times 10^3$	$3.2638 \times 10^3$	$2.8447 \times 10^1$
DE	$2.8874 \times 10^3$	$3.0941 \times 10^{-1}$	$4.6573 \times 10^3$	$7.3190 \times 10^1$	$3.2061 \times 10^3$	$3.4861 \times 10^0$
MFO	$3.2124 \times 10^3$	$3.9290 \times 10^2$	$5.8620 \times 10^3$	$4.3155 \times 10^2$	$3.2565 \times 10^3$	$2.4574 \times 10^1$
HGS	$2.8915 \times 10^3$	$1.3659 \times 10^1$	$4.9433 \times 10^3$	$3.3033 \times 10^2$	$3.2306 \times 10^3$	$1.5047 \times 10^1$
CPA	$2.8988 \times 10^3$	$1.8851 \times 10^1$	$4.3956 \times 10^3$	$1.0423 \times 10^3$	$3.2447 \times 10^3$	$2.3567 \times 10^1$
	F28		F29		F30	
	Avg	Std	Avg	Std	Avg	Std
OLCPA	$3.1166 \times 10^3$	$3.6219 \times 10^1$	$3.5490 \times 10^3$	$1.4289 \times 10^2$	$7.7032 \times 10^3$	$2.1026 \times 10^3$
CCMWOA	$4.5490 \times 10^3$	$5.1054 \times 10^2$	$5.3770 \times 10^3$	$7.8372 \times 10^2$	$7.2310 \times 10^7$	$6.3627 \times 10^7$
IGWO	$3.2621 \times 10^3$	$3.0754 \times 10^1$	$3.8055 \times 10^3$	$1.8503 \times 10^2$	$3.8641 \times 10^6$	$3.0498 \times 10^6$
CCMSCSA	$3.2293 \times 10^3$	$2.6311 \times 10^1$	$3.7148 \times 10^3$	$1.9800 \times 10^2$	$1.5266 \times 10^4$	$8.5355 \times 10^3$
BMWOA	$3.3944 \times 10^3$	$4.6263 \times 10^1$	$4.7907 \times 10^3$	$3.5255 \times 10^2$	$5.9215 \times 10^6$	$3.3072 \times 10^6$
CMFO	$3.3422 \times 10^3$	$5.8653 \times 10^1$	$4.5953 \times 10^3$	$3.6267 \times 10^2$	$1.8825 \times 10^6$	$5.1423 \times 10^6$
CESCA	$7.1979 \times 10^3$	$3.7582 \times 10^2$	$6.0902 \times 10^3$	$2.2499 \times 10^2$	$2.5420 \times 10^9$	$8.2166 \times 10^8$
GCHHO	$3.2262 \times 10^3$	$2.5508 \times 10^1$	$4.0266 \times 10^3$	$2.1016 \times 10^2$	$1.1463 \times 10^4$	$4.0577 \times 10^3$
DE	$3.1752 \times 10^3$	$4.9966 \times 10^1$	$3.5037 \times 10^3$	$6.6936 \times 10^1$	$1.3382 \times 10^4$	$3.7353 \times 10^3$
MFO	$4.5833 \times 10^3$	$9.6836 \times 10^2$	$4.2258 \times 10^3$	$2.8249 \times 10^2$	$9.2011 \times 10^5$	$1.1203 \times 10^6$
HGS	$3.2078 \times 10^3$	$5.5946 \times 10^1$	$3.7668 \times 10^3$	$1.5864 \times 10^2$	$9.8961 \times 10^4$	$1.2709 \times 10^5$
CPA	$3.1488 \times 10^3$	$5.0178 \times 10^1$	$3.7367 \times 10^3$	$1.9795 \times 10^2$	$1.1526 \times 10^4$	$4.4883 \times 10^3$
Overall rank						
	Rank	Avg	+ / = / -			
OLCPA	1	3.1322	~			
CCMSCSA	2	3.6022	15/8/7			
CPA	3	3.6933	15/13/2			
DE	4	3.7878	12/9/9			
HGS	5	4.6933	19/9/2			
IGWO	6	5.4800	18/8/4			
GCHHO	7	5.7567	24/0/0			
CMFO	8	8.2333	29/1/0			
MFO	9	8.2611	29/0/1			
BMWOA	10	9.0422	29/1/0			
CCMWOA	11	1.0409	30/0/0			
CESCA	12	1.1909	30/0/0			

From the experimental data in Table 5, we can see that OLCPA ranks first and has a smaller mean value of 3.1322 compared to the mean value of 3.6933 for CPA, which reflects the stronger effect of OLCPA than CPA. It can be seen that OLCPA outperformed CCMWOA and CESCA for all 30 functions on CEC 2017. Although the performance of CMSCSA is close to that of OLCPA, OLCPA performs better for multimodal and mixed modes, and OLCPA is the first among these competitors.

From the results of the Wilcoxon signed-rank test in Table 6, the *p*-values of most algorithms are less than 0.05, proving the statistically superior performance of OLCPA compared to other algorithms.



**Table 6.** The *p*-value of OLCPA and other algorithms.

	CCMWOA	IGWO	CCMSCSA	BMWOA	CMFO	CESCA
F1	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$7.1889 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F2	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F3	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F4	$1.7344 \times 10^{-6}$	$4.2857 \times 10^{-6}$	$4.7292 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F5	$1.7344 \times 10^{-6}$	$1.1748 \times 10^{-2}$	$1.1265 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F6	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F7	$1.7344 \times 10^{-6}$	$3.8811 \times 10^{-4}$	$2.8786 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F8	$1.7344 \times 10^{-6}$	$3.8723 \times 10^{-2}$	$8.9364 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$2.3534 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F9	$1.7344 \times 10^{-6}$	$4.4052 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$2.6033 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F10	$1.7344 \times 10^{-6}$	$3.4053 \times 10^{-5}$	$9.3157 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F11	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7138 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F12	$1.7344 \times 10^{-6}$	$1.9209 \times 10^{-6}$	$3.1603 \times 10^{-2}$	$1.7344 \times 10^{-6}$	$4.7292 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F13	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$3.3173 \times 10^{-4}$	$1.7344 \times 10^{-6}$	$6.3391 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F14	$1.7344 \times 10^{-6}$	$1.9209 \times 10^{-6}$	$8.4661 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F15	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$9.2626 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$2.1630 \times 10^{-5}$	$1.7344 \times 10^{-6}$
F16	$1.7344 \times 10^{-6}$	$5.8571 \times 10^{-1}$	$4.1653 \times 10^{-1}$	$3.5152 \times 10^{-6}$	$5.7924 \times 10^{-5}$	$1.7344 \times 10^{-6}$
F17	$1.7344 \times 10^{-6}$	$8.9364 \times 10^{-1}$	$3.8203 \times 10^{-1}$	$2.1266 \times 10^{-6}$	$9.3157 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F18	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F19	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$3.8723 \times 10^{-2}$	$1.7344 \times 10^{-6}$	$5.2872 \times 10^{-4}$	$1.7344 \times 10^{-6}$
F20	$1.7344 \times 10^{-6}$	$6.8836 \times 10^{-1}$	$8.6121 \times 10^{-1}$	$2.6033 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F21	$1.7344 \times 10^{-6}$	$2.4147 \times 10^{-3}$	$1.2381 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$2.1266 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F22	$1.2381 \times 10^{-5}$	$1.0201 \times 10^{-1}$	$2.5637 \times 10^{-2}$	$4.9916 \times 10^{-3}$	$2.3038 \times 10^{-2}$	$1.7344 \times 10^{-6}$
F23	$1.7344 \times 10^{-6}$	$1.5286 \times 10^{-1}$	$2.8486 \times 10^{-2}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F24	$6.9838 \times 10^{-6}$	$2.1266 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$2.9894 \times 10^{-1}$	$6.5833 \times 10^{-1}$	$1.7344 \times 10^{-6}$
F25	$1.7344 \times 10^{-6}$	$1.1499 \times 10^{-4}$	$9.7110 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F26	$1.7344 \times 10^{-6}$	$2.2102 \times 10^{-1}$	$9.8421 \times 10^{-3}$	$1.9729 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F27	$1.7344 \times 10^{-6}$	$5.0383 \times 10^{-1}$	$9.8421 \times 10^{-3}$	$3.5152 \times 10^{-6}$	$2.6033 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F28	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F29	$1.7344 \times 10^{-6}$	$1.4936 \times 10^{-5}$	$6.6392 \times 10^{-4}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
F30	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7988 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$
	GCHHO	DE	MFO	HGS	CPA	
F1	$4.1653 \times 10^{-1}$	$1.5886 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$8.1878 \times 10^{-5}$	$4.0702 \times 10^{-2}$	
F2	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.9209 \times 10^{-6}$	
F3	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	
F4	$3.4053 \times 10^{-5}$	$1.9729 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$1.1499 \times 10^{-4}$	$4.4493 \times 10^{-5}$	
F5	$3.8822 \times 10^{-6}$	$8.3071 \times 10^{-4}$	$9.3157 \times 10^{-6}$	$4.7795 \times 10^{-1}$	$4.5281 \times 10^{-1}$	
F6	$1.7344 \times 10^{-6}$	$4.3205 \times 10^{-8}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.0000 \times 10^0$	
F7	$1.7344 \times 10^{-6}$	$7.5213 \times 10^{-2}$	$1.7344 \times 10^{-6}$	$6.6392 \times 10^{-4}$	$1.4704 \times 10^{-1}$	
F8	$6.3391 \times 10^{-6}$	$1.3591 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$4.0483 \times 10^{-1}$	$6.2884 \times 10^{-1}$	
F9	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.6046 \times 10^{-4}$	$1.7518 \times 10^{-2}$	
F10	$2.8786 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7791 \times 10^{-1}$	$4.2843 \times 10^{-1}$	
F11	$1.3595 \times 10^{-4}$	$8.2206 \times 10^{-2}$	$1.7344 \times 10^{-6}$	$8.2167 \times 10^{-3}$	$3.0861 \times 10^{-1}$	
F12	$8.2206 \times 10^{-2}$	$3.4053 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$7.1889 \times 10^{-1}$	$1.2866 \times 10^{-3}$	
F13	$1.2866 \times 10^{-3}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.9209 \times 10^{-6}$	$5.3044 \times 10^{-1}$	
F14	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$2.1266 \times 10^{-6}$	$1.2381 \times 10^{-5}$	
F15	$1.9646 \times 10^{-3}$	$8.9187 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$1.2506 \times 10^{-4}$	$1.1093 \times 10^{-1}$	
F16	$5.7096 \times 10^{-2}$	$1.7344 \times 10^{-6}$	$1.0570 \times 10^{-4}$	$1.2044 \times 10^{-1}$	$2.1827 \times 10^{-2}$	
F17	$5.2872 \times 10^{-4}$	$1.3601 \times 10^{-5}$	$2.1266 \times 10^{-6}$	$4.6818 \times 10^{-3}$	$1.1748 \times 10^{-2}$	
F18	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.9209 \times 10^{-6}$	$4.7292 \times 10^{-6}$	
F19	$2.4308 \times 10^{-2}$	$1.3820 \times 10^{-3}$	$3.5152 \times 10^{-6}$	$1.1138 \times 10^{-3}$	$2.0671 \times 10^{-2}$	
F20	$4.0715 \times 10^{-5}$	$1.9209 \times 10^{-6}$	$1.9209 \times 10^{-6}$	$1.0357 \times 10^{-3}$	$1.8910 \times 10^{-4}$	
F21	$4.7292 \times 10^{-6}$	$2.9575 \times 10^{-3}$	$2.1266 \times 10^{-6}$	$4.4052 \times 10^{-1}$	$1.9861 \times 10^{-1}$	
F22	$3.8203 \times 10^{-1}$	$9.9179 \times 10^{-1}$	$1.1499 \times 10^{-4}$	$1.5886 \times 10^{-1}$	$3.4908 \times 10^{-1}$	
F23	$1.7344 \times 10^{-6}$	$5.5774 \times 10^{-1}$	$2.1266 \times 10^{-6}$	$1.4704 \times 10^{-1}$	$4.5281 \times 10^{-1}$	

Table 6. Cont.

	GCHHO	DE	MFO	HGS	CPA
F24	$1.7138 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$6.3391 \times 10^{-6}$	$2.5967 \times 10^{-5}$	$9.2710 \times 10^{-3}$
F25	$6.5641 \times 10^{-2}$	$6.1431 \times 10^{-1}$	$2.3534 \times 10^{-6}$	$9.5899 \times 10^{-1}$	$3.5009 \times 10^{-2}$
F26	$7.1570 \times 10^{-4}$	$9.0993 \times 10^{-1}$	$6.9838 \times 10^{-6}$	$4.9498 \times 10^{-2}$	$9.2626 \times 10^{-1}$
F27	$2.4147 \times 10^{-3}$	$1.9209 \times 10^{-6}$	$3.5009 \times 10^{-2}$	$1.7518 \times 10^{-2}$	$7.1889 \times 10^{-1}$
F28	$1.9209 \times 10^{-6}$	$3.7172 \times 10^{-5}$	$1.7344 \times 10^{-6}$	$1.6394 \times 10^{-5}$	$8.1574 \times 10^{-4}$
F29	$1.9209 \times 10^{-6}$	$1.5286 \times 10^{-1}$	$1.7344 \times 10^{-6}$	$8.1878 \times 10^{-5}$	$1.7088 \times 10^{-3}$
F30	$4.5336 \times 10^{-4}$	$3.5152 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$1.7344 \times 10^{-6}$	$2.5967 \times 10^{-5}$

Figure 4 depicts the convergence curves of OLCPA and the other algorithms, and it can be seen from these convergence function plots that OLCPA has the best ability to find the optimal solution compared to the other algorithms. Although the competition between HGS and OLCPA is fierce in functions F4, F12, and F19, OLCPA converges with increasing speed and accuracy in the later iterations and finally finds the optimal solution. Based on the trend of these convergence plots, it can be seen that the convergence speed and accuracy of OLCPA are better than those of the competitors.

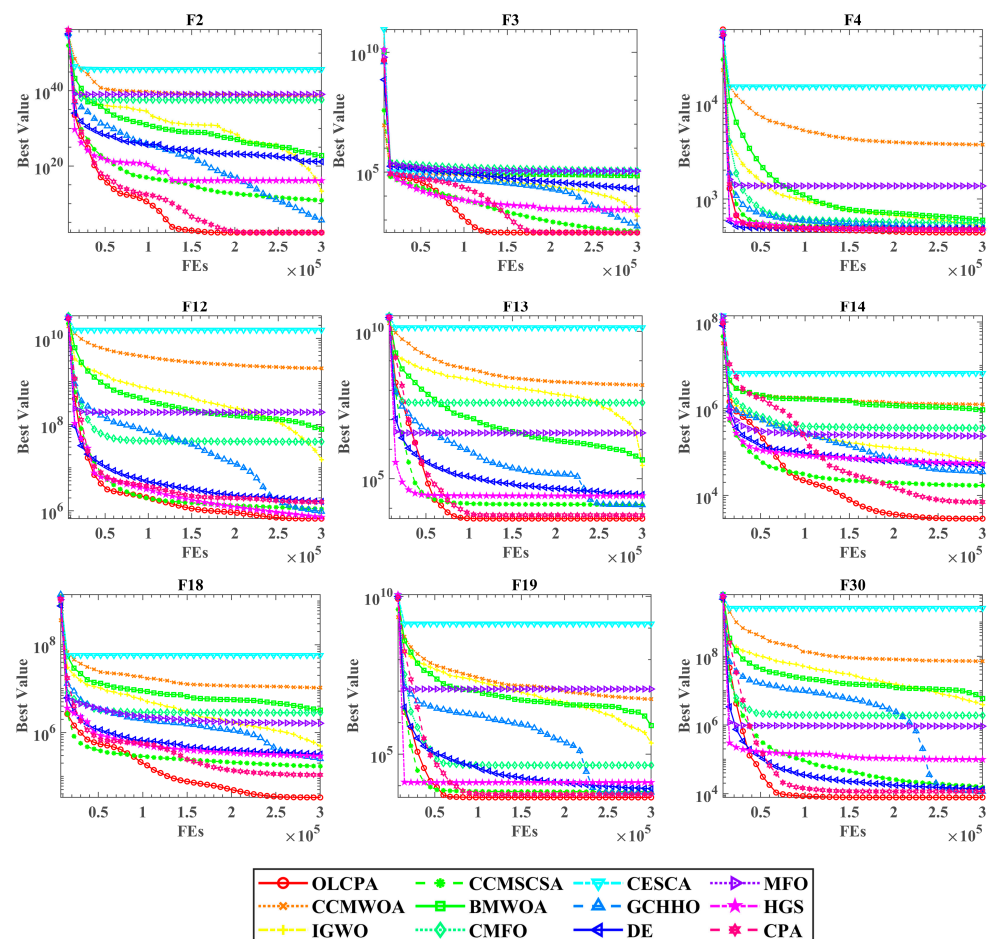


Figure 4. Convergence curves of OLCPA and other algorithms. In F4, OLCPA competes fiercely with HGS, but OLCPA finally finds the optimal value. In other benchmark functions, OLCPA’s convergence speed is relatively fast compared with the other algorithms. Additionally, with the increase in iterations, its exploration ability is enhanced, which means it can explore more valuable areas and finally find the optimal value.

## 6. Application in ECG Signal Classification

### 6.1. Test Datasets

This section focuses on the datasets used for training and testing. PhysioNet is a well-known research resource for studying complex physiological signals, including the MIT-BIH Arrhythmia Database [82] and the European ST-T Database [83], used for testing and training experiments.

#### 6.1.1. MIT-BIH Arrhythmia Database

Data in the MIT-BIH database were obtained from the ECG recordings of 47 patients, 60% of whom were inpatients and 40% of whom were outpatients, tested by the Boeheim Arrhythmia Laboratory from 1975 to 1979. These records consist of 48 half-hour dual-channel ECG recordings; each record was sampled at a rate of 360 Hz, and the two channels were modified limb leads (MLII) with V channels from V1 to V5.

#### 6.1.2. European ST-T Database

The European ST-T database contains mainly ST and t-wave variations, and the data are derived from 90 ECG recordings from 79 test subjects, who were men between 30 and 84 years of age and women between 55 and 71. Each recording spanned 2 h and contained two signals, each with a sampling rate of 250 Hz.

Due to the need to select an appropriate sample size, the MIT-BIH database was classified into four categories: N, VEBs, Q, and S, following the AAMI standard classification approach. The AAMI classification approach is described in Table 7. Due to the uneven sample distribution, four categories (N, S, Q, VEB) from the AMMI classification were selected for the MIT-BIH database for testing, and three (N, S, VEB) were selected for the European ST-T database.

**Table 7.** Relationship between AMMI and MIT-BIH in terms of categories.

AAMI Classes	Supraventricular Ectopic Beat (S)	Normal (N)	Ventricular Ectopic Beats (VEBs)	Unknown Beat (Q)	Fusion (F)
MIT-BIH classes	Aberrated atrial premature beat (a)	Normal beat (N)	Ventricular flutter wave (!)	Paced beat (/)	Fusion of ventricular and normal beat (F)
	Supraventricular premature beat (S)	Left bundle branch block beat (L)	Ventricular escape beat (E)	Unclassifiable beat (Q)	
	Atrial premature beat (A)	Right bundle branch block beat (R)	Premature ventricular contraction (V)		
	Nodal (junctional) premature beat (J)				
	Nodal (junctional) escape beat (j)				
	Atrial escape beat (e)				

In our experiment, the number of European ST-T datasets used is 3000, of which 2000 were in the training set and 1000 were in the test set, and the number of MIT-BIT datasets used is 10,000, of which 8000 were in the training set and 2000 were in the test set. Table 8 shows the number of samples per category for the datasets.

**Table 8.** Number of samples per category for the datasets.

	ST-T	MIT-BIH
N	1000	2500
S	1000	2500
VEB	1000	2500
Q	-	2500

### 6.2. Metrics for Performance Evaluation

The evaluation metrics used in this section differ from those used in Section 5. In Section 5, mean and standard deviation were utilized to evaluate the performance of the OLCPA algorithm, which is a continuous problem. However, this section mainly focuses on utilizing the OLCPA-CNN model to solve ECG electrocardiogram signal classification problems, which are categorical problems and differ from continuous problems. Therefore, in order to effectively evaluate the performance of the OLCPA-CNN model and compare it with methods used by other researchers, common metrics in deep learning such as accuracy (ACC), precision ( $Pr$ ), specificity ( $Sp$ ), sensitivity ( $Se$ ), and F-score (F1) are utilized. They are calculated as follows:

$$Se = \frac{TP}{TP + FN} \quad (18)$$

$$Sp = \frac{TN}{FP + TN} \quad (19)$$

$$Pr = \frac{TP}{TP + FP} \quad (20)$$

$$F1 = \frac{2TP}{2TP + FP + FN} \quad (21)$$

where the meanings represented by  $TP$ ,  $TN$ ,  $FP$ , and  $FN$  are described in Section 5.3.

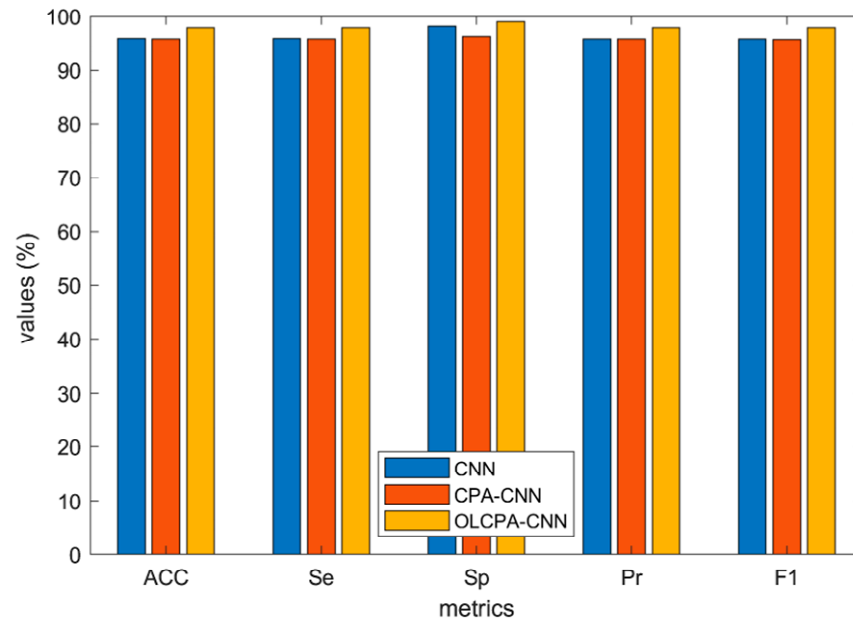
ACC indicates the rate of correct classification. The value of ACC can reflect the good or bad performance of the model classification; when the value of the former is large, accordingly, the performance of the latter is also good. The sensitivity indicates the number of positive samples as a percentage of the number of all true positive samples. The sensitivity value is proportional to the classification accuracy of positive samples, and when the value of the former increases, the latter also increases accordingly. Specificity indicates the number of samples predicted to be negative among all true negative samples. Precision indicates the proportion of true positive samples among all positive samples. The F1 integrally evaluates the performance of a classifier, and the larger the F1, the better the performance of this classifier.

### 6.3. Performance Analysis of OLCPA-CNN on Datasets

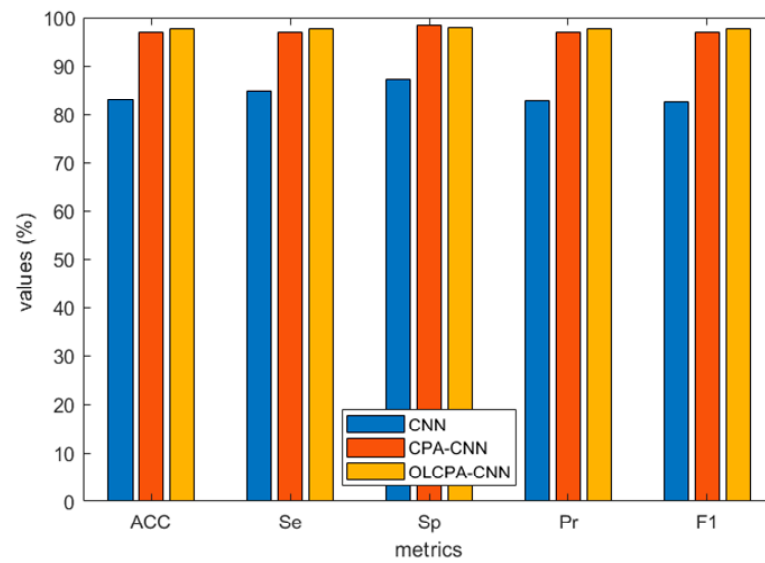
The proposed OLCPA-CNN model was tested and trained using two different datasets. Since OLCPA can tune the parameters of the CNN through the iterative update of the population, it allows the CNN to play a better role in classification. The classification accuracy of this model on the two datasets (MIT-BIH, ST-T) is 97.9% and 97.8%, respectively, which shows that the performance of this OLCPA-CNN model is excellent.

Figure 5 depicts the average values of the evaluation metrics for OLCPA-CNN, CPA-CNN, and a randomly generated CNN on the MIT-BIH dataset; the results in the figure indicate that the evaluation metrics of OLCPA-CNN are higher than those of CNN and CPA-CNN, which indicates that OLCPA-CNN outperforms CPA-CNN. Figure 6 shows the average performance of OLCPA-CNN, the randomly generated CNN, and CPA-CNN on the ST-T dataset. The results in the figure indicate that the performance of OLCPA-CNN is significantly stronger than that of CPA-CNN and the other CNN, which reflects that the overall optimization of OLCPA-CNN is good. Figures 7 and 8 show the correct and loss rates of OLCPA-CNN on the ST-T and MIT-BIH datasets, respectively, and the trend of the

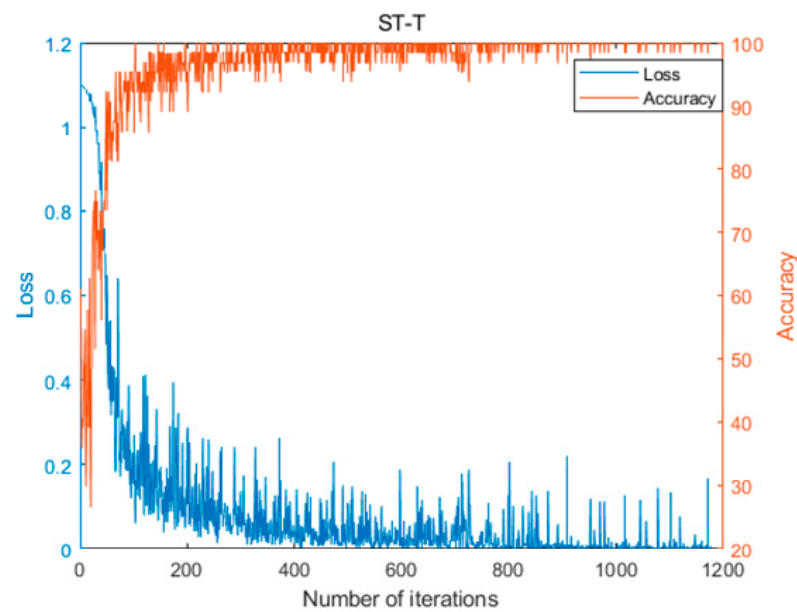
curves shows that as the number of iterations increases, the correct rate approaches 100% and the loss rate converges to 0.



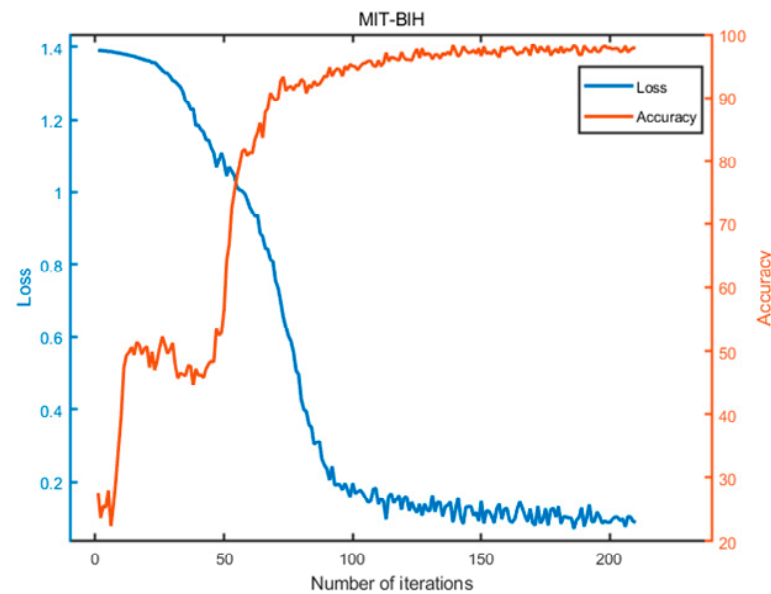
**Figure 5.** Overview of the average performance metrics of OLCPA-CNN, CPA-CNN, and the randomly generated CNN on the MIT-BIH dataset.



**Figure 6.** Overview of the average performance metrics of OLCPA-CNN, CPA-CNN, and the randomly generated CNN on the ST-T dataset.



**Figure 7.** The accuracy and loss curve of OLCPA-CNN on the ST-T dataset.



**Figure 8.** The accuracy and loss of OLCPA-CNN on the MIT-BIH dataset.

To highlight the effective performance of OLCPA-CNN on the MIT-BIH dataset for classification performance, OLCPA-CNN was compared with other methods. These methods include the following: Li et al. [84] proposed a model for optimizing support vector machine classifiers using a genetic algorithm and used it for the MIT-BIH dataset; Patro et al. [85] proposed optimizing machine learning using optimization algorithms and applied this to the MIT-BIH dataset, where the algorithms and classifiers involved include the support vector machine (SVM) and random forest (RF), genetic algorithm (GA) and particle swarm algorithm (PSO); Acharya et al. [29] investigated a nine-layer convolutional neural network structure for identifying five different classes of heartbeats in ECG signals. The experimental data in Table 9 reflect that OLCPA-CNN is effective compared with these other methods.

**Table 9.** Comparison of OLCPA-CNN with other methods on the MIT-BIH dataset.

Reference	Method	Acc	Se
Acharya et al. [54]	CNN	94.03%	96.71%
Li et al. [84]	SVM	97.30%	97.40%
Patro et al. [85]	PSO, GA, SVM, and RF	95.30%	94.00%
Proposed	OLCPA-CNN	97.90%	97.90%

## 7. Discussion

The classification results demonstrate that OLCPA-CNN can automatically search for the best hyperparameters suitable for a CNN, and the proposed OLCPA-CNN model can effectively address ECG classification tasks. Moreover, the ability to automatically extract features and perform classification is precious, particularly considering the significant expense associated with manual feature annotation by specialized professionals. However, when optimizing complex network architectures and handling vast amounts of data, this technique also has the drawback of high time costs. In addition, there is currently no universally applicable solution to every problem, and it is necessary to decide on the optimization algorithm and network architecture—including hyperparameters, number of neurons, and layers—to use based on the specific problem being addressed. Therefore, to solve ECG classification problems, this study selected an improved CPA algorithm, namely the OLCPA optimization algorithm, to optimize the hyperparameters of a CNN. Of course, in future work, it can also be applied to more cases, such as the optimization of machine learning models [86], fine-grained alignment [87], computational experiments [88,89], Alzheimer’s disease identification [90], iris or retinal vessel segmentation [91,92], MRI reconstruction [93], service ecosystem [94], structured sparsity optimization [95], tensor recovery [96,97], medical image computing [98], computer-aided medical diagnosis [99], image denoising [100], renewable energy generation [101], and medical signals [102,103].

## 8. Conclusions and Future Works

This article presents an OLCPA algorithm based on orthogonal learning strategies and proposes an OLCPA-CNN model that optimizes the hyperparameters of a CNN using the OLCPA algorithm. The experimental results show that the OLCPA-CNN model achieves excellent classification performance, with an accuracy of 97.90% on the MIT-BIH dataset, outperforming other models proposed by researchers. As the MIT-BIH dataset is a type of time-series data, the OLCPA-CNN model presented in this paper can be used for ECG classification and other time-series datasets, such as geomagnetic data. However, the use of this model is limited and it may not be suitable for different classification problems. Therefore, specific algorithms should be designed, and appropriate network structures should be optimized for specific problems.

In future work, reducing the running time will be the focus due to the drawback of this parameter optimization process being a waste of time. Secondly, this model is also valuable for other problems, such as regression problems. Third, CPA can be combined with other network structures.

**Declaration of AI and AI-Assisted Technologies in the Writing Process:** During the revision of this work, the authors used ChatGPT in order to enhance the English grammar and paraphrase some sentences. After using this tool/service, the authors reviewed and edited the content as needed, and they takes full responsibility for the content of the publication.

**Author Contributions:** X.H.: Conceptualization, Software, Data Curation; W.S.: Conceptualization, Investigation, Writing—Original Draft, Project Administration; R.Z.: Software, Formal Analysis, Data Curation; A.A.H.: Methodology, Writing—Original Draft, Writing—Review and Editing; H.C.: Methodology, Validation, Investigation, Supervision; Y.Z.: Validation, Formal Analysis, Writing—Review and Editing, Funding Acquisition. All authors have read and agreed to the published version of the manuscript.



**Funding:** This work was supported by the Natural Science Foundation of Hebei Province (D2022512001); the National Natural Science Foundation of China (42164002); MRC, UK (MC\_PC\_17171); Royal Society, UK (RP202G0230); BHF, UK (AA/18/3/34220); Hope Foundation for Cancer Research, UK (RM60G0680); GCRF, UK (P202PF11); Sino-UK Industrial Fund, UK (RP202G0289); LIAS, UK (P202ED10, P202RE969); Data Science Enhancement Fund, UK (P202RE237); Fight for Sight, UK (24NN201); Sino-UK Education Fund, UK (OP202006); BBSRC, UK (RM32G0178B8).

**Data Availability Statement:** The numerical and experimental data used to support the findings of this study are included within the article.

**Conflicts of Interest:** The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

## References

1. Li, L.; Wang, P.; Zheng, X.; Xie, Q.; Tao, X.; Velásquez, J.D. Dual-interactive fusion for code-mixed deep representation learning in tag recommendation. *Inf. Fusion* **2023**, *99*, 101862. [[CrossRef](#)]
2. Liu, H.; Yue, Y.; Liu, C.; Spencer Jr, B.; Cui, J. Automatic recognition and localization of underground pipelines in GPR B-scans using a deep learning model. *Tunn. Undergr. Space Technol.* **2023**, *134*, 104861. [[CrossRef](#)]
3. Zhao, K.; Jia, Z.; Jia, F.; Shao, H. Multi-scale integrated deep self-attention network for predicting remaining useful life of aero-engine. *Eng. Appl. Artif. Intell.* **2023**, *120*, 105860. [[CrossRef](#)]
4. Deng, Y.; Lv, J.; Huang, D.; Du, S. Combining the theoretical bound and deep adversarial network for machinery open-set diagnosis transfer. *Neurocomputing* **2023**, *548*, 126391. [[CrossRef](#)]
5. Deng, X.; Liu, E.; Li, S.; Duan, Y.; Xu, M. Interpretable Multi-modal Image Registration Network Based on Disentangled Convolutional Sparse Coding. *IEEE Trans. Image Process.* **2023**, *32*, 1078–1091. [[CrossRef](#)] [[PubMed](#)]
6. Guan, Z.; Jing, J.; Deng, X.; Xu, M.; Jiang, L.; Zhang, Z.; Li, Y. DeepMIH: Deep invertible network for multiple image hiding. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 372–390. [[CrossRef](#)]
7. Xu, J.; Pan, S.; Sun, P.Z.H.; Park, S.H.; Guo, K. Human-Factors-in-Driving-Loop: Driver Identification and Verification via a Deep Learning Approach using Psychological Behavioral Data. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 3383–3394. [[CrossRef](#)]
8. Lu, H.; Zhu, Y.; Yin, M.; Yin, G.; Xie, L. Multimodal fusion convolutional neural network with cross-attention mechanism for internal defect detection of magnetic tile. *IEEE Access* **2022**, *10*, 60876–60886. [[CrossRef](#)]
9. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]
10. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [[CrossRef](#)]
11. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
12. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
13. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
14. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
15. Cao, B.; Zhao, J.; Gu, Y.; Fan, S.; Yang, P. Security-aware industrial wireless sensor network deployment optimization. *IEEE Trans. Ind. Inform.* **2019**, *16*, 5309–5316. [[CrossRef](#)]
16. Zhang, K.; Wang, Z.; Chen, G.; Zhang, L.; Yang, Y.; Yao, C.; Wang, J.; Yao, J. Training effective deep reinforcement learning agents for real-time life-cycle production optimization. *J. Pet. Sci. Eng.* **2022**, *208*, 109766. [[CrossRef](#)]
17. Cao, B.; Zhao, J.; Gu, Y.; Ling, Y.; Ma, X. Applying graph-based differential grouping for multiobjective large-scale optimization. *Swarm Evol. Comput.* **2020**, *53*, 100626. [[CrossRef](#)]
18. Cao, B.; Zhao, J.; Lv, Z.; Gu, Y.; Yang, P.; Halgamuge, S.K. Multiobjective Evolution of Fuzzy Rough Neural Network via Distributed Parallelism for Stock Prediction. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 939–952. [[CrossRef](#)]
19. Cao, B.; Li, M.; Liu, X.; Zhao, J.; Cao, W.; Lv, Z. Many-Objective Deployment Optimization for a Drone-Assisted Camera Network. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 2756–2764. [[CrossRef](#)]
20. Cao, B.; Zhao, J.; Yang, P.; Gu, Y.; Muhammad, K.; Rodrigues, J.J.; de Albuquerque, V.H.C. Multiobjective 3-D topology optimization of next-generation wireless data center network. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3597–3605. [[CrossRef](#)]
21. Cao, B.; Fan, S.; Zhao, J.; Tian, S.; Zheng, Z.; Yan, Y.; Yang, P. Large-scale many-objective deployment optimization of edge servers. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3841–3849. [[CrossRef](#)]
22. Zhang, J.; Tang, Y.; Wang, H.; Xu, K. ASRO-DIO: Active Subspace Random Optimization Based Depth Inertial Odometry. *IEEE Trans. Robot.* **2022**, *39*, 1496–1508. [[CrossRef](#)]
23. Li, R.; Wu, X.; Tian, H.; Yu, N.; Wang, C. Hybrid Memetic Pretrained Factor Analysis-Based Deep Belief Networks for Transient Electromagnetic Inversion. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5920120. [[CrossRef](#)]

24. Shan, W.; He, X.; Liu, H.; Heidari, A.A.; Wang, M.; Cai, Z.; Chen, H. Cauchy mutation boosted Harris hawk algorithm: Optimal performance design and engineering applications. *J. Comput. Des. Eng.* **2023**, *10*, 503–526. [[CrossRef](#)]
25. Shan, W.; Qiao, Z.; Heidari, A.A.; Chen, H.; Turabieh, H.; Teng, Y. Double adaptive weights for stabilization of moth flame optimizer: Balance analysis, engineering cases, and medical diagnosis. *Knowl.-Based Syst.* **2021**, *214*, 106728. [[CrossRef](#)]
26. Tian, J.; Hou, M.; Bian, H.; Li, J. Variable surrogate model-based particle swarm optimization for high-dimensional expensive problems. *Complex Intell. Syst.* **2022**, 1–49. [[CrossRef](#)]
27. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2011**, *29*, 17–35. [[CrossRef](#)]
28. Chen, H.; Li, C.; Mafarja, M.; Heidari, A.A.; Chen, Y.; Cai, Z. Slime mould algorithm: A comprehensive review of recent variants and applications. *Int. J. Syst. Sci.* **2022**, *54*, 204–235. [[CrossRef](#)]
29. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [[CrossRef](#)]
30. Yang, Y.; Chen, H.; Heidari, A.A.; Gandomi, A.H. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. *Expert Syst. Appl.* **2021**, *177*, 114864. [[CrossRef](#)]
31. Ahmadianfar, I.; Asghar Heidari, A.; Gandomi, A.H.; Chu, X.; Chen, H. RUN Beyond the Metaphor: An Efficient Optimization Algorithm Based on Runge Kutta Method. *Expert Syst. Appl.* **2021**, *181*, 115079. [[CrossRef](#)]
32. Ahmadianfar, I.; Asghar Heidari, A.; Noshadian, S.; Chen, H.; Gandomi, A.H. INFO: An Efficient Optimization Algorithm based on Weighted Mean of Vectors. *Expert Syst. Appl.* **2022**, *195*, 116516. [[CrossRef](#)]
33. Tu, J.; Chen, H.; Wang, M.; Gandomi, A.H. The Colony Predation Algorithm. *J. Bionic Eng.* **2021**, *18*, 674–710. [[CrossRef](#)]
34. Su, H.; Zhao, D.; Asghar Heidari, A.; Liu, L.; Zhang, X.; Mafarja, M.; Chen, H. RIME: A physics-based optimization. *Neurocomputing* **2023**, *532*, 183–214. [[CrossRef](#)]
35. Zhang, Y.; Liu, R.; Heidari, A.A.; Wang, X.; Chen, Y.; Wang, M.; Chen, H. Towards augmented kernel extreme learning models for bankruptcy prediction: Algorithmic behavior and comprehensive analysis. *Neurocomputing* **2021**, *430*, 185–212. [[CrossRef](#)]
36. Dong, R.; Chen, H.; Heidari, A.A.; Turabieh, H.; Mafarja, M.; Wang, S. Boosted kernel search: Framework, analysis and case studies on the economic emission dispatch problem. *Knowl.-Based Syst.* **2021**, *233*, 107529. [[CrossRef](#)]
37. Xue, Y.; Cai, X.; Neri, F. A multi-objective evolutionary algorithm with interval based initialization and self-adaptive crossover operator for large-scale feature selection in classification. *Appl. Soft Comput.* **2022**, *127*, 109420. [[CrossRef](#)]
38. Hu, H.; Shan, W.; Chen, J.; Xing, L.; Heidari, A.A.; Chen, H.; He, X.; Wang, M. Dynamic Individual Selection and Crossover Boosted Forensic-based Investigation Algorithm for Global Optimization and Feature Selection. *J. Bionic Eng.* **2023**, 1–27. [[CrossRef](#)]
39. Liang, J.; Qiao, K.; Yu, K.; Qu, B.; Yue, C.; Guo, W.; Wang, L. Utilizing the Relationship between Unconstrained and Constrained Pareto Fronts for Constrained Multiobjective Optimization. *IEEE Trans. Cybern.* **2022**, *53*, 3873–3886. [[CrossRef](#)]
40. Yu, K.; Zhang, D.; Liang, J.; Chen, K.; Yue, C.; Qiao, K.; Wang, L. A Correlation-Guided Layered Prediction Approach for Evolutionary Dynamic Multiobjective Optimization. *IEEE Trans. Evol. Comput.* **2022**, *1*. [[CrossRef](#)]
41. Deng, W.; Xu, J.; Gao, X.Z.; Zhao, H. An Enhanced MSIQDE Algorithm with Novel Multiple Strategies for Global Optimization Problems. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 1578–1587. [[CrossRef](#)]
42. Chen, J.; Cai, Z.; Chen, H.; Chen, X.; Escorcia-Gutierrez, J.; Mansour, R.F.; Ragab, M. Renal Pathology Images Segmentation Based on Improved Cuckoo Search with Diffusion Mechanism and Adaptive Beta-Hill Climbing. *J. Bionic Eng.* **2023**, 1–36. [[CrossRef](#)]
43. Xue, Y.; Tong, Y.; Neri, F. An ensemble of differential evolution and Adam for training feed-forward neural networks. *Inf. Sci.* **2022**, *608*, 453–471. [[CrossRef](#)]
44. Wen, X.; Wang, K.; Li, H.; Sun, H.; Wang, H.; Jin, L. A two-stage solution method based on NSGA-II for Green Multi-Objective integrated process planning and scheduling in a battery packaging machinery workshop. *Swarm Evol. Comput.* **2021**, *61*, 100820. [[CrossRef](#)]
45. Huang, C.; Zhou, X.; Ran, X.; Liu, Y.; Deng, W.; Deng, W. Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem. *Inf. Sci.* **2023**, *619*, 2–18. [[CrossRef](#)]
46. Zhao, C.; Zhou, Y.; Lai, X. An integrated framework with evolutionary algorithm for multi-scenario multi-objective optimization problems. *Inf. Sci.* **2022**, *600*, 342–361. [[CrossRef](#)]
47. Li, C.; Sun, G.; Deng, L.; Qiao, L.; Yang, G. A population state evaluation-based improvement framework for differential evolution. *Inf. Sci.* **2023**, *629*, 15–38. [[CrossRef](#)]
48. Pyakillya, B.; Kazachenko, N.; Mikhailovsky, N. Deep Learning for ECG Classification. *J. Phys. Conf. Ser.* **2017**, *913*, 012004. [[CrossRef](#)]
49. Mathews, S.M.; Kambhamettu, C.; Barner, K.E. A novel application of deep learning for single-lead ECG classification. *Comput. Biol. Med.* **2018**, *99*, 53–62. [[CrossRef](#)]
50. Sannino, G.; De Pietro, G. A deep learning approach for ECG-based heartbeat classification for arrhythmia detection. *Future Gener. Comput. Syst.* **2018**, *86*, 446–455. [[CrossRef](#)]
51. Strodthoff, N.; Wagner, P.; Schaeffter, T.; Samek, W. Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL. *IEEE J. Biomed. Health Inf.* **2021**, *25*, 1519–1528. [[CrossRef](#)]
52. Peimankar, A.; Puthusserypady, S. DENS-ECG: A deep learning approach for ECG signal delineation. *Expert Syst. Appl.* **2021**, *165*, 113911. [[CrossRef](#)]

53. Hasan, N.I.; Bhattacharjee, A. Deep Learning Approach to Cardiovascular Disease Classification Employing Modified ECG Signal from Empirical Mode Decomposition. *Biomed. Signal Process. Control* **2019**, *52*, 128–140. [[CrossRef](#)]
54. Acharya, U.R.; Oh, S.L.; Hagiwara, Y.; Tan, J.H.; Adam, M.; Gertych, A.; San Tan, R. A deep convolutional neural network model to classify heartbeats. *Comput. Biol. Med.* **2017**, *89*, 389–396. [[CrossRef](#)]
55. Houssein, E.H.; Hassaballah, M.; Ibrahim, I.E.; AbdElminaam, D.S.; Wazery, Y.M. An automatic arrhythmia classification model based on improved Marine Predators Algorithm and Convolutions Neural Networks. *Expert Syst. Appl.* **2022**, *187*, 115936. [[CrossRef](#)]
56. Khalifa, M.H.; Ammar, M.; Ouarda, W.; Alimi, A.M. Particle swarm optimization for deep learning of convolution neural network. In Proceedings of the 2017 Sudan Conference on Computer Science and Information Technology (SCCSIT), Elnuhood, Sudan, 17–19 November 2017; pp. 1–5.
57. Yamasaki, T.; Honma, T.; Aizawa, K. Efficient optimization of convolutional neural networks using particle swarm optimization. In Proceedings of the 2017 IEEE Third International Conference on Multimedia Big Data (BigMM), Laguna Hills, CA, USA, 19–21 April 2017; pp. 70–73.
58. Dey, S.; Roychoudhury, R.; Malakar, S.; Sarkar, R. An optimized fuzzy ensemble of convolutional neural networks for detecting tuberculosis from Chest X-ray images. *Appl. Soft Comput.* **2022**, *114*, 108094. [[CrossRef](#)]
59. Pathan, S.; Siddalingaswamy, P.C.; Ali, T. Automated Detection of Covid-19 from Chest X-ray scans using an optimized CNN architecture. *Appl. Soft Comput.* **2021**, *104*, 107238. [[CrossRef](#)]
60. Ezzat, D.; Hassanien, A.E.; Ella, H.A. An optimized deep learning architecture for the diagnosis of COVID-19 disease based on gravitational search optimization. *Appl. Soft Comput.* **2021**, *98*, 106742. [[CrossRef](#)] [[PubMed](#)]
61. Singh, P.; Chaudhury, S.; Panigrahi, B.K. Hybrid MPSO-CNN: Multi-level Particle Swarm optimized hyperparameters of Convolutional Neural Network. *Swarm Evol. Comput.* **2021**, *63*, 100863. [[CrossRef](#)]
62. Fernandes Junior, F.E.; Yen, G.G. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm Evol. Comput.* **2019**, *49*, 62–74. [[CrossRef](#)]
63. Fisher, R.A. A Mathematical Examination of the Methods of Determining the Accuracy of an Observation by the Mean Error, and by the Mean Square Error. *Mon. Not. R. Astron. Soc.* **1920**, *80*, 758–770. [[CrossRef](#)]
64. Zhuang, Y.; Chen, S.; Jiang, N.; Hu, H. An Effective WSENet-Based Similarity Retrieval Method of Large Lung CT Image Databases. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 2359–2376.
65. Huang, C.-Q.; Jiang, F.; Huang, Q.-H.; Wang, X.-Z.; Han, Z.-M.; Huang, W.-Y. Dual-graph attention convolution network for 3-D point cloud classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–13. [[CrossRef](#)]
66. Guo, F.; Zhou, W.; Lu, Q.; Zhang, C. Path extension similarity link prediction method based on matrix algebra in directed networks. *Comput. Commun.* **2022**, *187*, 83–92. [[CrossRef](#)]
67. Liu, X.; He, J.; Liu, M.; Yin, Z.; Yin, L.; Zheng, W. A Scenario-Generic Neural Machine Translation Data Augmentation Method. *Electronics* **2023**, *12*, 2320. [[CrossRef](#)]
68. Cheng, L.; Yin, F.; Theodoridis, S.; Chatzis, S.; Chang, T.-H. Rethinking Bayesian learning for data analysis: The art of prior and inference in sparsity-aware modeling. *IEEE Signal Process. Mag.* **2022**, *39*, 18–52. [[CrossRef](#)]
69. Song, X.; Tong, W.; Lei, C.; Huang, J.; Fan, X.; Zhai, G.; Zhou, H. A clinical decision model based on machine learning for ptosis. *BMC Ophthalmol.* **2021**, *21*, 169. [[CrossRef](#)] [[PubMed](#)]
70. Xie, X.; Huang, L.; Marson, S.M.; Wei, G. Emergency response process for sudden rainstorm and flooding: Scenario deduction and Bayesian network analysis using evidence theory and knowledge meta-theory. *Nat. Hazards* **2023**, 1–23. [[CrossRef](#)]
71. Wang, S.; Hu, X.; Sun, J.; Liu, J. Hyperspectral anomaly detection using ensemble and robust collaborative representation. *Inf. Sci.* **2023**, *624*, 748–760. [[CrossRef](#)]
72. Wu, G.; Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization*; Technical Report; National University of Defense Technology: Changsha, China; Kyungpook National University: Daegu, Republic of Korea; Nanyang Technological University: Singapore, 2017.
73. Luo, J.; Chen, H.; Heidari, A.A.; Xu, Y.; Zhang, Q.; Li, C. Multi-strategy boosted mutative whale-inspired optimization approaches. *Appl. Math. Model.* **2019**, *73*, 109–123. [[CrossRef](#)]
74. Long, W.; Liang, X.; Cai, S.; Jiao, J.; Zhang, W. A modified augmented Lagrangian with improved grey wolf optimization to constrained optimization problems. *Neural Comput. Appl.* **2017**, *28*, 421–438. [[CrossRef](#)]
75. Shan, W.; Hu, H.; Cai, Z.; Chen, H.; Liu, H.; Wang, M.; Teng, Y. Multi-strategies Boosted Mutative Crow Search Algorithm for Global Tasks: Cases of Continuous and Discrete Optimization. *J. Bionic Eng.* **2022**, *19*, 1830–1849. [[CrossRef](#)]
76. Heidari, A.A.; Aljarah, I.; Faris, H.; Chen, H.; Luo, J.; Mirjalili, S. An enhanced associative learning-based exploratory whale optimizer for global optimization. *Neural Comput. Appl.* **2020**, *32*, 5185–5211. [[CrossRef](#)]
77. Wang, M.; Chen, H.; Yang, B.; Zhao, X.; Hu, L.; Cai, Z.; Huang, H.; Tong, C. Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. *Neurocomputing* **2017**, *267*, 69–84. [[CrossRef](#)]
78. Lin, A.; Wu, Q.; Heidari, A.A.; Xu, Y.; Chen, H.; Geng, W.; Li, Y.; Li, C. Predicting Intentions of Students for Master Programs Using a Chaos-Induced Sine Cosine-Based Fuzzy K-Nearest Neighbor Classifier. *IEEE Access* **2019**, *7*, 67235–67248. [[CrossRef](#)]
79. Song, S.; Wang, P.; Heidari, A.A.; Wang, M.; Zhao, X.; Chen, H.; He, W.; Xu, S. Dimension decided Harris hawks optimization with Gaussian mutation: Balance analysis and diversity patterns. *Knowl.-Based Syst.* **2021**, *215*, 106425. [[CrossRef](#)]



80. Price, K.V. Differential Evolution. In *Handbook of Optimization*; Springer: Berlin/Heidelberg, Germany, 2013. [[CrossRef](#)]
81. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [[CrossRef](#)]
82. Moody, G.B.; Mark, R.G. The impact of the MIT-BIH arrhythmia database. *IEEE Eng. Med. Biol. Mag.* **2001**, *20*, 45–50. [[CrossRef](#)]
83. Taddei, A.; Distanti, G.; Emdin, M.; Pisani, P.; Moody, G.B.; Zeelenberg, C.; Marchesi, C. The European ST-T database: Standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography. *Eur. Heart J.* **1992**, *13*, 1164–1172. [[CrossRef](#)]
84. Li, H.; Yuan, D.; Wang, Y.; Cui, D.; Cao, L. Arrhythmia classification based on multi-domain feature extraction for an ECG recognition system. *Sensors* **2016**, *16*, 1744. [[CrossRef](#)]
85. Patro, K.K.; Jaya Prakash, A.; Jayamanmadha Rao, M.; Rajesh Kumar, P. An Efficient Optimized Feature Selection with Machine Learning Approach for ECG Biometric Recognition. *IETE J. Res.* **2020**, *68*, 2743–2754. [[CrossRef](#)]
86. Zhao, C.; Wang, H.; Chen, H.; Shi, W.; Feng, Y. JAMNet: A Remote Pulse Extraction Network Based on Joint Attention and Multi-Scale Fusion. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *33*, 2783–2797. [[CrossRef](#)]
87. Wang, S.; Wang, B.; Zhang, Z.; Heidari, A.A.; Chen, H. Class-aware sample reweighting optimal transport for multi-source domain adaptation. *Neurocomputing* **2023**, *523*, 213–223. [[CrossRef](#)]
88. Xue, X.; Yu, X.-N.; Zhou, D.-Y.; Wang, X.; Zhou, Z.-B.; Wang, F.-Y. Computational Experiments: Past, Present and Future. *arXiv* **2022**, arXiv:2202.13690.
89. Xue, X.; Yu, X.; Zhou, D.; Peng, C.; Wang, X.; Liu, D.; Wang, F.-Y. Computational Experiments for Complex Social Systems—Part III: The Docking of Domain Models. *IEEE Trans. Comput. Soc. Syst.* **2023**, 1–15. [[CrossRef](#)]
90. Yan, B.; Li, Y.; Li, L.; Yang, X.; Li, T.-q.; Yang, G.; Jiang, M. Quantifying the impact of Pyramid Squeeze Attention mechanism and filtering approaches on Alzheimer’s disease classification. *Comput. Biol. Med.* **2022**, *148*, 105944. [[CrossRef](#)]
91. Chen, Y.; Gan, H.; Chen, H.; Zeng, Y.; Xu, L.; Heidari, A.A.; Zhu, X.; Liu, Y. Accurate iris segmentation and recognition using an end-to-end unified framework based on MADNet and DSANet. *Neurocomputing* **2023**, *517*, 264–278. [[CrossRef](#)]
92. Li, Y.; Zhang, Y.; Cui, W.; Lei, B.; Kuang, X.; Zhang, T. Dual Encoder-Based Dynamic-Channel Graph Convolutional Network with Edge Enhancement for Retinal Vessel Segmentation. *IEEE Trans. Med. Imaging* **2022**, *41*, 1975–1989. [[CrossRef](#)]
93. Lv, J.; Li, G.; Tong, X.; Chen, W.; Huang, J.; Wang, C.; Yang, G. Transfer learning enhanced generative adversarial networks for multi-channel MRI reconstruction. *Comput. Biol. Med.* **2021**, *134*, 104504. [[CrossRef](#)] [[PubMed](#)]
94. Xue, X.; Li, G.; Zhou, D.; Zhang, Y.; Zhang, L.; Zhao, Y.; Feng, Z.; Cui, L.; Zhou, Z.; Sun, X. Research Roadmap of Service Ecosystems: A Crowd Intelligence Perspective. *Int. J. Crowd Sci.* **2022**, *6*, 195–222. [[CrossRef](#)]
95. Zhang, X.; Zheng, J.; Wang, D.; Tang, G.; Zhou, Z.; Lin, Z. Structured Sparsity Optimization with Non-Convex Surrogates of  $\ell_{2,0}$ -Norm: A Unified Algorithmic Framework. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 6386–6402. [[CrossRef](#)]
96. Zhang, X.; Wang, D.; Zhou, Z.; Ma, Y. Robust Low-Rank Tensor Recovery with Rectification and Alignment. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 238–255. [[CrossRef](#)]
97. Zhang, X.; Zheng, J.; Zhao, L.; Zhou, Z.; Lin, Z. Tensor Recovery with Weighted Tensor Average Rank. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–15. [[CrossRef](#)]
98. Nabavi, S.; Ejmalian, A.; Moghaddam, M.E.; Abin, A.A.; Frangi, A.F.; Mohammadi, M.; Rad, H.S. Medical imaging and computational image analysis in COVID-19 diagnosis: A review. *Comput. Biol. Med.* **2021**, *135*, 104605. [[CrossRef](#)] [[PubMed](#)]
99. Faruqui, N.; Yousuf, M.A.; Whaiduzzaman, M.; Azad, A.K.M.; Barros, A.; Moni, M.A. LungNet: A hybrid deep-CNN model for lung cancer diagnosis using CT and wearable sensor-based medical IoT data. *Comput. Biol. Med.* **2021**, *139*, 104961. [[CrossRef](#)] [[PubMed](#)]
100. Zhang, X.; Zheng, J.; Wang, D.; Zhao, L. Exemplar-Based Denoising: A Unified Low-Rank Recovery Framework. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 2538–2549. [[CrossRef](#)]
101. Sun, X.; Cao, X.; Zeng, B.; Zhai, Q.; Guan, X. Multistage Dynamic Planning of Integrated Hydrogen-Electrical Microgrids under Multiscale Uncertainties. *IEEE Trans. Smart Grid* **2022**, *1*. [[CrossRef](#)]
102. Dai, Y.; Wu, J.; Fan, Y.; Wang, J.; Niu, J.; Gu, F.; Shen, S. MSEva: A musculoskeletal rehabilitation evaluation system based on EMG signals. *ACM Trans. Sens. Netw.* **2022**, *19*, 1–23. [[CrossRef](#)]
103. Zhou, J.; Zhang, X.; Jiang, Z. Recognition of Imbalanced Epileptic EEG Signals by a Graph-Based Extreme Learning Machine. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 5871684. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.