Mathematical and Computational Applications

MDPI

Article

Comparative Study of Metaheuristic Optimization of Convolutional Neural Networks Applied to Face Mask Classification

Patricia Melin, Daniela Sánchez, Martha Pulido and Oscar Castillo *

Tijuana Institute of Technology, TecNM, Calzada Tecnológico S/N, Fracc. Tomas Aquino, Tijuana CP 22379, BC, Mexico; pmelin@tectijuana.mx (P.M.); daniela.sanchez@tectijuana.edu.mx (D.S.); martha.pulido@tectijuana.edu.mx (M.P.)
* Correspondence: ocastillo@tectijuana.mx

Abstract: The preventive measures taken to curb the spread of COVID-19 have emphasized the importance of wearing face masks to prevent potential infection with serious diseases during daily activities or for medical professionals working in hospitals. Due to the mandatory use of face masks, various methods employing artificial intelligence and deep learning have emerged to detect whether individuals are wearing masks. In this paper, we utilized convolutional neural networks (CNNs) to classify the use of face masks into three categories: no mask, incorrect mask, and proper mask. Establishing the appropriate CNN architecture can be a demanding task. This study compares four swarm intelligent metaheuristics: particle swarm optimization (PSO), grey wolf optimizer (GWO), bat algorithm (BA), and whale optimization algorithm (WOA). The CNN architecture design involves determining the essential hyperparameters of the CNNs. The results indicate the effectiveness of the PSO and BA in achieving an accuracy of 100% when using 10% of the images for testing. Meanwhile, when 90% of the images were used for testing, the results were as follows: PSO 97.15%, WOA 97.14%, BA 97.23%, and GWO 97.18%. These statistically significant differences demonstrate that the BA allows better results than the other metaheuristics analyzed in this study.

Keywords: face mask classification; swarm intelligence metaheuristics; convolutional neural network; particle swarm optimization; whale optimization algorithm; bat algorithm; grey wolf optimizer

Citation: Melin, P.; Sánchez, D.; Pulido, M.; Castillo, O. Comparative Study of Metaheuristic Optimization of Convolutional Neural Networks Applied to Face Mask Classification. *Math. Comput. Appl.* **2023**, *28*, 107. https://doi.org/10.3390/mca28060107

Academic Editor: Oliver Schütze

Received: 18 August 2023
Revised: 27 October 2023
Accepted: 27 October 2023
Published: 1 November 2023

1. Introduction

The COVID-19 pandemic has shown the importance of using face masks, avoiding the spread of the virus, and preventing the infection of millions of people [1,2]. However, it is important to mention that various studies on its use were performed several years before the COVID-19 pandemic, where the importance and efficacy of its use to prevent other respiratory infections were demonstrated [3,4]. Two of the most widely used subsets of artificial intelligence related to face masks are deep learning (DL) and machine learning (ML). Different works on the detection of the facial mask using pre-trained models of convolutional neural networks can be found in [5–7], which allowed us to observe the potential of this technique in the detection and classification of facial masks [8–10]. In Ref. [11], the authors studied the architectures of different pre-trained models such as EfficientNet, InceptionV3, MobileNetV1, MobileNetV2, ResNet-101, ResNet-50, VGG16, and VGG19. Based on their study, they proposed a model for face mask detection based on MobileNetV2, applying data augmentation techniques to increase the number of images for the training phase. In Ref. [12], an application for mobile devices was developed to identify face masks using the Google Cloud ML API, while analyzing the progress of cloud technology and the benefits of machine learning. In Ref. [10], the authors developed five ML models for face mask classification. The developed models were Naïve Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and K-Nearest Neighbors (KNN).

The test of the models was performed using 1222 images, where the results demonstrated the effectiveness of the DT over the other models. The use of neural networks is related to metaheuristics, which are utilized to find the optimal architectures that improve the results depending on the application for which the network is used [13]. Metaheuristics are a great option for finding optimal parameters in applications in different areas. These algorithms have been classified according to their inspiration: based on evolutionary algorithms, physics-based algorithms, and algorithms based on swarm intelligence [14–16]. Nature-inspired algorithms are mainly inspired by collective behavior, where the main characteristics of a particular species are analyzed and represented in a computational way to be used in solving complex problems in the search for optimal solutions [17,18]. In recent works, comparisons have been made between metaheuristics to compare the performances applied to find CNN hyperparameters. Some of these metaheuristics are grey wolf optimizer (GWO), whale optimization algorithm (WOA), salp swarm algorithm (SSA), sine cosine algorithm (SCA), multiverse optimizer (MVO), particle swarm optimization (PSO), moth flame optimization (MFO), and bat algorithm (BA), to mention a few. The authors have concluded the advantages of combining convolutional neural networks and metaheuristics for the search of hyperparameters [19–21]. These techniques have been combined to solve applications related to pattern recognition [19,22,23], image classifications [18,24], and medical diagnosis [21,25,26], among other applications.

In this work, convolutional neural network hyperparameters are optimized by different nature-inspired algorithms [27–29]. The optimized hyperparameters are the number of convolutional layers, filters, fully connected layers, neurons, batch size, and epochs. The contribution of this work includes the optimal design of the convolutional neural network architectures to increase classification accuracy and its application to face mask classification: no mask, incorrect mask, and mask. Recent works applied to face mask classification based their model architectures on pre-trained models, which does not guarantee optimal architecture. As a novelty, this paper proposed optimizing CNN architectures instead of basing them on other architectures. The optimal hyperparameters are found using four metaheuristics used in recent works to make a statistical comparison and analysis, providing better accuracy for face mask classification.

This paper is presented as follows. In Section 2, the metaheuristics applied in this work are presented in a succinct manner. Section 3 shows the optimization proposed for the convolutional neural networks. The results obtained by each swarm intelligence metaheuristic are shown in Section 4. The statistical test results are shown in Section 5. The conclusions are presented in Section 6.

2. Background

The optimal design of architectures and models has allowed the realization of important practical applications. In Ref. [30], optimal convolutional neural network architectures was designed to identify various types of damage on reinforced concrete (RC) to avoid further structure deterioration. The results achieved show good accuracy of six types of damage. The design of convolutional neural network architectures using a particle swarm optimization algorithm was proposed and applied to sign language recognition using three study cases of sign language databases: the Mexican Sign Language alphabet, the American Sign Language MNIST, and the American Sign Language alphabet [31]. In Ref. [32], the authors proposed face detection and face classification by developing adaptive sailfish moth flame optimization (ASMFO) to the parameter optimization using a deep learning approach. In Ref. [19], the authors analyzed the importance of the CNN hyperparameters, such as filters, kernel, epoch, batch size, and pooling size of the convolutional neural networks applied to classify human movements. They compared seven metaheuristic algorithms: GWO, WOA, SSA, SCA, MVO, PSO, and MFO, concluding the advantages of the metaheuristics to optimize the hyperparameters of CNNs. The results led the authors to the conclusion that the implementation of GWO achieved higher accuracy than the other metaheuristics. In Ref. [20], the authors proposed a PSO to determine optimal hyperpa-

rameters of convolutional neural networks. They used the simplest CNN model as a base: LeNet. Their results achieved better results when the PSO designed the CNN architectures. The results achieved by their study were obtained using MNIST, Fashion-MNIST, and CIFAR-10 datasets.

In previous works [33,34], nature-inspired algorithms have optimized modular neural network architectures applied to human recognition using different biometric measures. In those works, comparisons using genetic algorithms (GAs) and swarm intelligence algorithms were performed, and significant evidence of the advantage of the swarm intelligence algorithms was proven. More recently, in [22], and based on the advantages offered by the swarm intelligence algorithms, the architecture of convolutional neural networks was optimized and applied to face recognition. In this work, algorithms such as particle swarm optimization and grey wolf optimizer offer advantages when designing convolutional neural network architectures. It is important to mention that the databases used for this work were small, with 400 and 165 images. In Ref. [35], the non-optimized design of convolutional neural network architectures applied to the facial mask classification was performed, and the best architecture was implemented in a real-time system using a Raspberry Pi 4 in combination with a camera to obtain the image in real time. The Raspberry Pi 4 sends a signal through its GPIO Board, and a result is provided by lighting an LED. If the mask is correctly used, the green LED is turned on. If the mask is incorrectly used, the yellow LED is turned on, and if a mask is not used, the red LED is turned on.

3. Intelligence Techniques

This section shows a description of the intelligence techniques utilized in this work.

3.1. Convolutional Neural Networks

Artificial neural networks (ANNs) are mainly based on the behavior of the human nervous system and its way of processing information. An artificial neural network is a type of distributive processor made up of simple processing units known as neurons, simulating two main aspects of the human brain: it acquires knowledge of its environment through a learning process and the use of synaptic weights to store the required knowledge [36,37]. Learning methods are categorized into supervised, semi-supervised, and unsupervised learning. Among the main properties that can be found in ANNs that make them one of the main techniques used in artificial intelligence, we can find their capacity for generalization, adaptation, learning, and parallelism [38,39]. Convolutional neural networks (CNNs) are an improvement of ANNs with some characteristics that make them powerful in applications where images are used. This type of network consists of other layers in addition to those already existing in conventional neural networks: the convolutional and the pooling layers. One of the advantages provided by this type of network is the extraction of features from the given images before proceeding to the learning phase, which makes it possible to reduce the amount of information that must be learned by the ANN [38]. In the convolutional layers (CLs), the inputs are multiplied by a filter with the size $m \times n$. Each layer contains a height, width, and depth. When talking about depth concerning the layer, it refers to the number of channels (primary colors) that contain the input images [40]. The most used grouping layers with the maximum, average, and minimum are responsible for grouping the feature map produced in the convolution layer, thus reducing the amount of information that will pass to the fully connected layers [23,41]. In Figure 1, a representation of a convolutional neural network is shown.

3.2. Nature-Inspired Algorithms

The nature-inspired algorithms used in this study are described below.

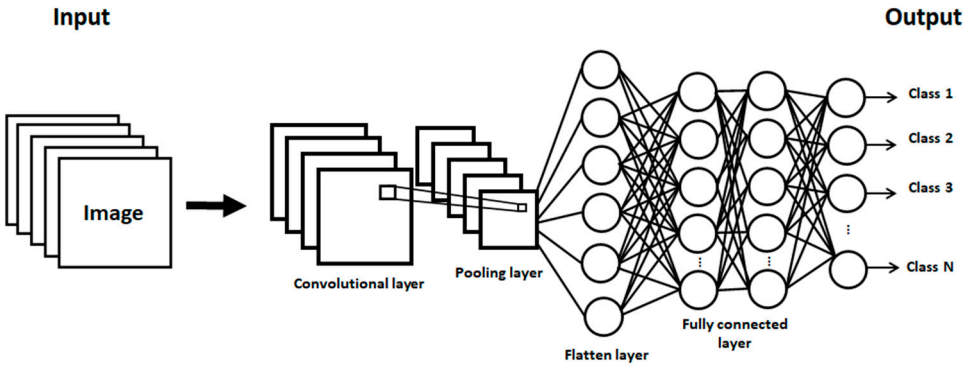

Figure 1. Representation of the architecture of a convolutional neural network.

3.2.1. Particle Swarm Optimization

In Ref. [42], the particle swarm optimization (PSO) based on the fish or bird social performance was proposed. A set of particles is known as a swarm, and each particle is a solution [43]. A particle defines their next position by Equation (1).

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{1}$$

where $x_{id}(t)$ indicates at time t, in the dimension d, the actual position of the particle i. A velocity $v_i(t+1)$ is designated to establish the next position. In Ref. [44], this algorithm was enhanced by adding the parameter: inertia weight (w). The particle velocity is defined by Equation (2).

$$v_{id}(t+1) = w \times v_{id}(t) + c_1 \times r_{1d}(t) \times [y_{id}(t) - x_{id}(t)] + c_2 \times r_{2d}(t) \times [\hat{y}_d(t) - x_{id}(t)] \tag{2}$$

where r_1 and r_2 are random values in [0, 1]. The best position of a particle i in dimension d is connoted by $y_{id}(t)$; the best position of the swarm in d dimension is denoted by $\hat{y}_d(t)$. c_1 and c_2 are the cognitive and social components.

w has a decreased value during the algorithm execution to allow exploitation and exploration. The linear decrease in the inertia weight applied in this work is given by Equation (3).

$$w_t = (w_s - w_e) \times \frac{(t_{max} - t)}{t_{max}} + w_e \tag{3}$$

where t_{max} denotes the maximum number of time steps, and w_s and w_e are the initial and final values of the inertia weight, respectively. The recommended values are w_s = 0.9 and w_e = 0.4 [45].

3.2.2. Grey Wolf Optimizer

In Ref. [46], the grey wolf optimizer (GWO) was proposed. This metaheuristic uses a dominant hierarchy applied by the wolves in hunting as inspiration. This dominant hierarchy is shown in Figure 2, where leaders known as alphas are at the top of the pyramid, and they make the main hunting and sleeping decisions. The betas are subaltern wolves that help the alpha wolves in making decisions. A delta wolf does not belong to any level already mentioned and can dominate only the lowest level. The delta wolves have different roles as scouts, sentinels, elders, hunters, and caretakers. The wolves in the lowest level are known as the omegas. They are always submitted by the wolves that are in the superior hierarchies [47,48].

The description of the principles used to define this algorithm and its mathematical representation is described below:

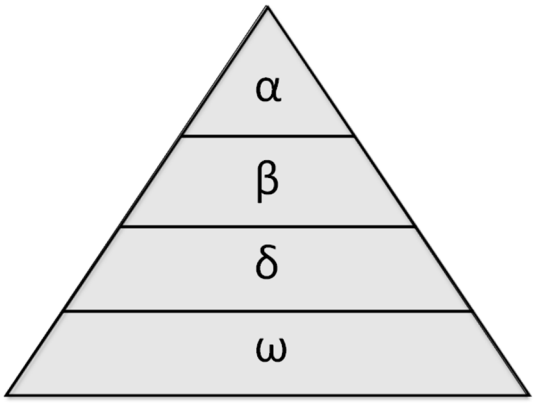

Figure 2. The dominance hierarchy of wolves.

- **Social hierarchy:** The three best solutions are alpha (α), beta (β), and delta (δ). The wolves belonging to the lowest level are the omegas (ω).
- **Encircling prey:** The process of prey encircling during hunting are represented by Equations (4) and (5).

$$\vec{D} = | \vec{C} \times \vec{X_p}\,(t) - \vec{X}\,(t) \tag{4}$$

$$\vec{X}(t+1) = \vec{X_p}(t) - \vec{A} \times \vec{D} \tag{5}$$

where $\vec{X}$ denotes the agent position in the t iteration, and $\vec{X_p}$ represents the position of the prey. The coefficient vectors are $\vec{A}$ and $\vec{C}$. Equations (6) and (7) are used to determine their values.

$$\vec{A} = 2\vec{a} \times \vec{r_1} - \vec{a} \tag{6}$$

$$\vec{C} = 2 \times \vec{r_2} \tag{7}$$

where $\vec{r_1}$ and $\vec{r_2}$ represent vectors with random values in [0, 1]. During the algorithm execution, the vector $\vec{a}$ has linear decreasing values in [2, 0] given by Equation (8) [49].

$$\vec{a}\,(t) = 2 - \frac{2 \times \mathrm{t}}{t_{max}} \tag{8}$$

where t denotes the current iteration, and t_{max} denotes the maximum number of iterations.

- **Hunting:** The first three levels in the dominant hierarchy know the prey position. With their positions, the wolves belonging to the lowest level (omega) can update their position using Equations (9)–(11).

$$\vec{D_\alpha} = \left| \vec{C_1} \times \vec{X_\alpha} - \vec{X} \right|, \quad \vec{D_\beta} = \left| \vec{C_2} \times \vec{X_\beta} - \vec{X} \right|, \quad \vec{D_\delta} = \left| \vec{C_3} \times \vec{X_\delta} - \vec{X} \right| \tag{9}$$

$$\vec{X_1} = \vec{X_\alpha} - \vec{A_1} \times \left(\vec{D_\alpha} \right), \quad \vec{X_2} = \vec{X_\beta} - \vec{A_2} \times \left(\vec{D_\beta} \right), \quad \vec{X_3} = \vec{X_\delta} - \vec{A_3} \times \left(\vec{D_\delta} \right) \tag{10}$$

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \tag{11}$$

- **Attacking prey:** The process is also known as exploitation, where the current position of an agent and the prey allows it to establish the next position of the agent. This position is calculated using $\vec{a}$ and vector $\vec{A}$ with random values in an interval $[-2a,\ 2a]$.
- **Search for prey:** The process is also known as exploration, where vector $\vec{C}$ is used with values in [0, 2] to provide diversity to the population and avoid local optimal.

3.2.3. Whale Optimization Algorithm

In Ref. [50], the whale optimization algorithm (WOA) was proposed. This algorithm uses as inspiration the hunting method applied by the whales. These marine mammals usually live in groups and are considered killers and predators [51]. One of the main characteristics shared with the grey wolf optimizer is the process of encircling prey, present also in WOA. The description of the processes used to define this algorithm based on humpback whales and its mathematical representation is described below:

- **Encircling prey:** The whales encircle the prey because they know its position. The whale closest to the prey becomes the best solution. Equations (3) and (4) allow the update of the position of the rest of the agents.
- **Bubble-net attacking method:** This process is also known as exploitation and is very similar to the one in the GWO, where the distance between the agent and the prey is determined. The process can be accomplished using two approaches:
  1. **Mechanism of shrinking encircling:** In Equation (5), the values of $\vec{a}$ decrease every iteration, and an interval $[-a,\ a]$ is used to generate random values for the vector $\vec{A}$.
  2. **Spiral updating position:** The helix-shaped movement of whales between the whale and prey position is mimicked by Equation (12).

$$\vec{X}(t+1) = \vec{D'} \times e^{bl} \times \cos(2\pi l) + \vec{X_p}(t) \tag{12}$$

where the distance between prey and whale is connoted by $\vec{D'}$, and b is a constant that represents the shape of the logarithmic spiral. A random value in an interval $[-1, 1]$ is represented by l.

- **Search for prey:** This process is also known as exploration, where the whales seek randomly based on the position of others. To force the exploration, the $\vec{A}$ vector has numbers less than -1 and greater than 1. The process is defined by Equations (13) and (14).

$$\vec{D} = \left| \vec{C} \times \vec{X_{rand}} - \vec{X} \right| \tag{13}$$

$$\vec{X}(t+1) = \vec{X_{rand}} - \vec{A} \times \vec{D} \tag{14}$$

where a random whale of the current iteration is represented by $\vec{X_{rand}}$. This random whale or the best solution found is utilized to help the other whales update their position.

3.2.4. Bat Algorithm

The bat algorithm (BA) [52] is based on their echolocation behavior due to the ability they have to identify their prey even in darkness. There are different types of bats depending on their size. Microbats have the characteristic of using a type of sonar known as echolocation, allowing them to detect prey and avoid obstacles. The bats make a loud sound pulse and listen for the echo that is reflected off of nearby objects. The rate of pulse is established in an interval [0, 1] [53].

The author established some important rules to delimit the behavior and knowledge that the bats can have:

- Echolocation is used for all the bats to sense distance, and they know the difference between the prey and other kind of elements.
- To search for prey, each bat flies randomly in a position x_i with a velocity v_i. This task is performed by changing loudness A and wavelength λ. Depending on the closeness of its objective, the bat regulates the wavelength of its emitted pulses and regulate the rate of pulse emission $r \in [0, 1]$.
- The loudness is assumed to be a large value positive number A to a minimum constant value A_{min}.

To define the update of position and velocities, the next equations are given by Equations (15)–(17).

$$f_i = f_{min} + (f_{max} + f_{min}) \times \beta \tag{15}$$

$$v_i(t) = v_i(t-1) + (x_i(t-1) - x_*) \times f_i \tag{16}$$

$$x_i(t) = v_i(t-1) + v_i(t) \tag{17}$$

where $x_i(t)$ and $v_i(t)$ represent the new position and velocity, respectively, at time step t. A vector with random values in [0, 1] is represented by β. The current global best solution is denoted by x_*. For the local search, the best solutions are used to select one of them and locally generate a new solution using a random walk given by Equation (18) [54].

$$x_{new} = x_{old} + \varepsilon \times A(t) \tag{18}$$

where ε represents a random value in an interval $[-1, 1]$, and the average loudness of all the bats at time step t is represented by $A(t)$.

4. Proposed Method

The proposed optimization is applied to face mask classification (no mask, incorrect mask, and mask). To perform this task, the method combines CNNs and optimization algorithms. The metaheuristics allow the optimal design of CNN architectures to be found. The optimization algorithm designs the CNN architectures, seeking their number of convolutional layers, filters, fully connected layers, neurons, batch size, and epochs. Each CL is followed by a max-pooling layer with a pool size of 3×3 to reduce image size. Figure 3 shows an example of the CNN architecture applied to face mask classification. As input to the convolutional neural network, images of people wearing (correctly and incorrectly) or not using face masks are used for the training phase of the convolutional neural network. The first layers of the convolutional neural network (convolutional and pooling) will extract features and reduce the image so that the fully connected layers learn the most relevant information. As output, when an image is simulated, a classification will be obtained (no mask, incorrect mask, and mask). A correct classification will depend on correct learning and the convolutional neural network architecture. For this reason, an optimization algorithm is an excellent option for designing the architecture because it allows a specific model to be applied to a particular application.

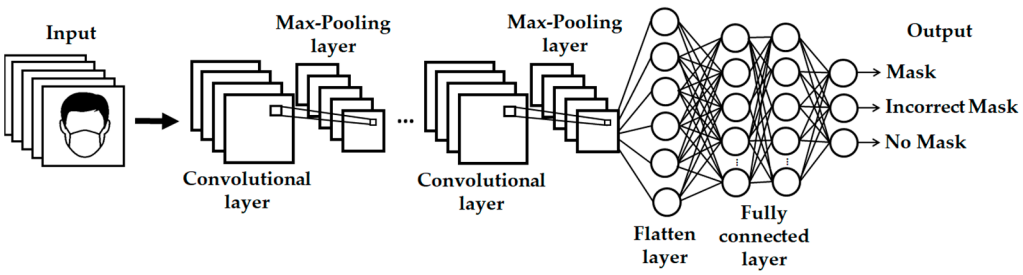

Figure 3. Illustration of the CNN architecture applied to face mask classification.

4.1. Description of the Optimization

The parameters used to execute any optimization algorithm have great importance because these depend on its performance. For each optimization algorithm, 10 solutions (particles, bats, or search agents) and 10 iterations are used. The configuration of the optimization algorithms used in this work is presented in more detail in Table 1. The parameters presented are based on previous works [22,33,55].

Table 1. Configuration of characteristic/tuning parameters of the optimization algorithms.

PSO		**BAT**		**WOA and GWO**	
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
Particles	10	Bats	10	Search Agents	10
Maximum Iterations (t_{max})	10	Maximum Iterations (t_{max})	10	Maximum Iterations (t_{max})	10
C_1	2	f_{min}	0	-	-
C_2	2	f_{max}	2	-	-
w_s	0.9	Loudness (A)	0.5	-	-
w_e	0.4	Pulse rate (r)	0.5	-	-

Each solution seeks to minimize the face mask classification error. In this work, the accuracy equation is used and given by Equation (19).

$$Accuraccy = \frac{TP + TN}{TP + FP + TN + FN} \tag{19}$$

where TP, TN, FP, and FN mean True Positive, True Negative, False Positive, and False Negative, respectively. The objective function used in this work is expressed by Equation (20).

$$f = 1 - \frac{TP + TN}{TP + FP + TN + FN} \tag{20}$$

The search space used for each solution (particle or agent) is determined by the minimum and maximum ranges shown in Table 2. These ranges are established based on previous works [18,22]. The convolutional neural networks use the Adaptive Moment Estimation (Adam) as a leaning algorithm and the rectified linear activation function (ReLU) as an activation function. The batch size is determined using a range from 1 up to 5, which means 8, 16, 32, 64, or 128.

Each particle or agent represents a solution, where each solution has 14 dimensions, which allow the creation of a CNN. In Figure 4, the dimensions of the solution are shown. The first four dimensions allow the determination of the number of convolutional layers, epoch, batch size, and the number of fully connected layers. Meanwhile, the rest of the dimensions allow us to determine the number of neurons and filters.

All the metaheuristics have, as a stopping criterion, 10 iterations or when the best solution has a cost equal to zero. The Keras Python package based on TensorFlow was used to implement the optimization algorithms and to build and train the CNN models.

Table 2. Definition of the search space to determine the solutions.

Hyperparameter		Minimum	Maximum
Convolutional layers (CLs)		1	5
Number of filters	CL 1	8	16
	CL 2	8	16
	CL 3	16	32
	CL 4	16	32
	CL 5	32	64
Fully connected layers (FCL)		1	5
Neurons		10	150
Epoch		5	50
Batch Size		1	5

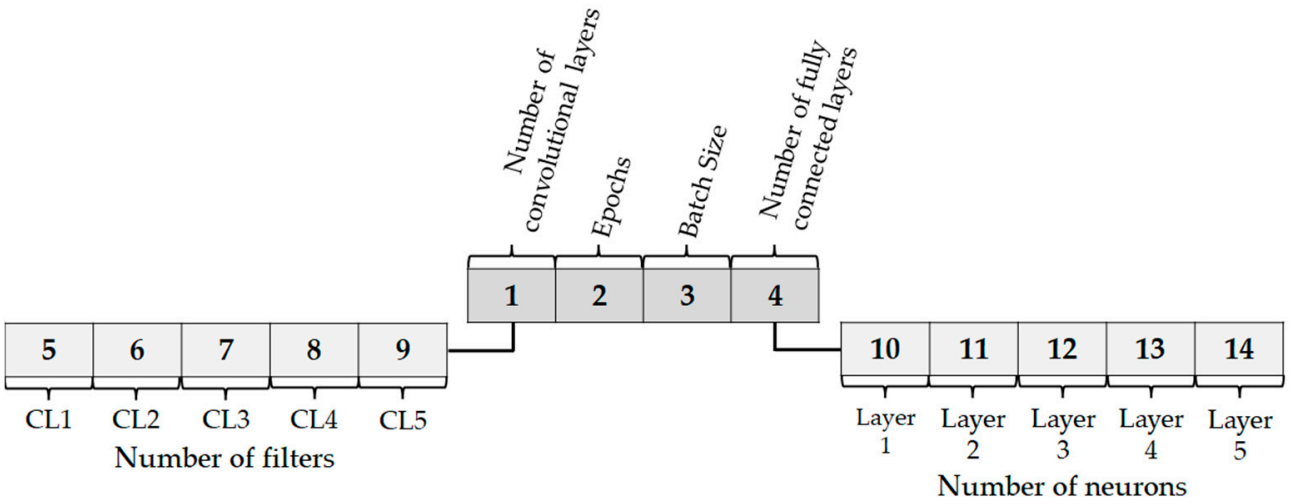

Figure 4. Dimensions of the solutions to design CNN architectures. CL indicates convolutional layer, and Layer indicates fully connected layer.

4.2. Database

To perform the face mask classification, the convolution neural networks are trained, validated, and tested using images of three classes (no mask, incorrect mask, and mask). The first two classes are obtained from the MaskedFace-Net dataset [56], and the no mask class is obtained from the Flickr-Faces-HQ Dataset (FFHQ) [57]. The MaskedFace-Net dataset consists of 137,016 images, and it is based on the Flickr-Faces-HQ (FFHQ) dataset. In this work, 3000 images were used, where each class contains 1000 images of the dataset. In Figure 5, a sample of the dataset is shown. The images used in this work were separated into training, validation, and testing. To help prevent bias in our models, when the images are split into sets, stratified sampling is utilized to guarantee a consistent distribution. Stratified sampling is a functionality provided by the Keras Python package.

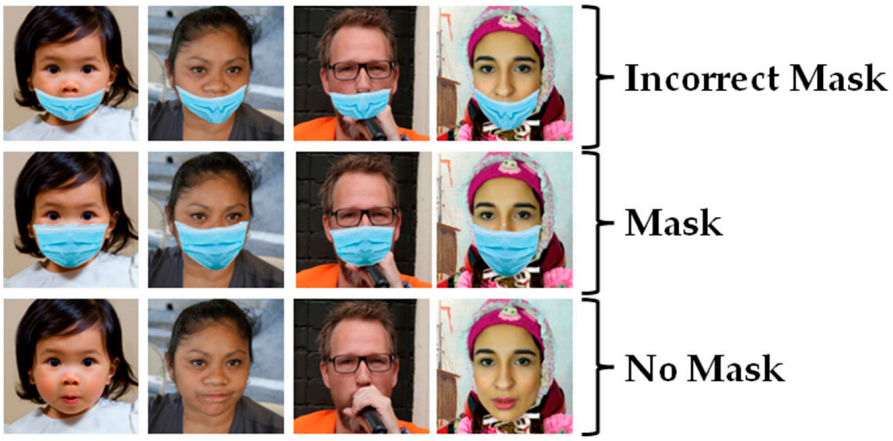

Figure 5. Examples of the database with 3 classes: incorrect, mask, and no mask.

4.3. Preprocessing

The original images have a resolution of 1024 × 1024 pixels. The region of interest (ROI) for this work is the face region, and it is automatically found using the Caffe model. The Caffe model was developed by the Berkeley Vision and Learning Center (BVLC). This model was trained to perform object detection and classification [58]. In Figure 6, an example of the face detection is shown.

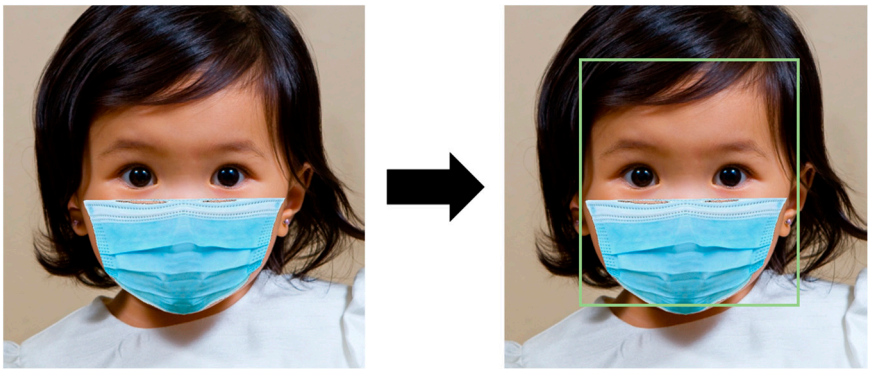

Figure 6. Application of the Caffe Model to detect the region of interest (face detection).

When the face region is detected, the image is resized to 100 × 100 pixels. Once the image is resized, an RGB subtraction technique is implemented to the ROI in order to help counteract slight variations [59]. In Figure 7, an example of the RGB subtraction technique is shown.

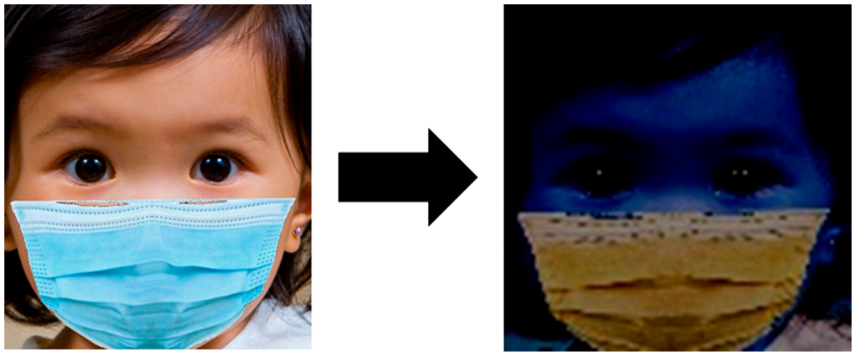

Figure 7. Example of the RGB subtraction applied to the ROI.

The proposed method is shown in Figure 8, which begins with the input images that go through preprocessing. The database is partitioned into three sets (training, validation, and testing), and the optimized CNN architecture is obtained with the metaheuristic.

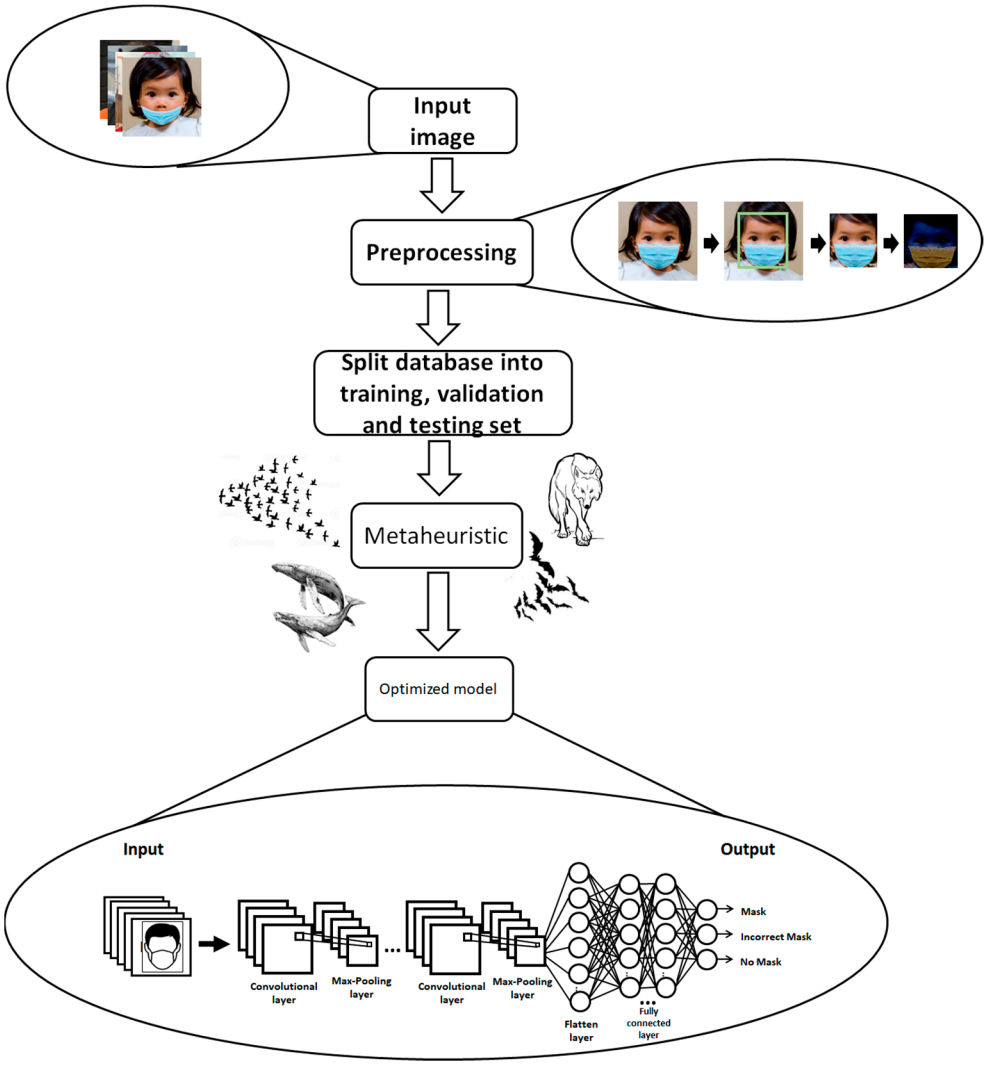

Figure 8. The flowchart of the proposed method begins with the input images up to the optimized CNN architecture using a metaheuristic.

5. Experimental Results

The database previously described is used to prove the proposed hyperparameters optimization. As previously mentioned, 3000 images were used to train, validate and test each convolutional neural network. In this work, 20 runs were performed using 10, 20, 30, 40, 50, 70, 80, and 90 percent of the images for testing, leaving the rest for training and validation. These experiments are performed with all the previously mentioned metaheuristics.

5.1. PSO Results

The best architectures achieved by the PSO with different percentages of images for the testing phase are summarized in Table 3. The best results are obtained with 10% and 20% of images for the testing phase, where an accuracy of 100% is achieved (marked with bold text in Table 3). We can define the best architecture as the one that uses less information for the training phase, which would be when 20% is used for the testing phase. This CNN model is structured as follows: four convolutional layers with 16, 16, 28, and 23 filters, with a size of 3×3. This architecture uses four FCLs with 150, 10, 117, and 19 neurons and a batch size of 8 with 12 epochs.

Table 3. The best accuracy results and architectures obtained by the PSO. CLs indicates the number of convolutional layers with their number of filters, and FCLs indicates the number of fully connected layers with their number of neurons.

% Images for Testing	CLs (Filters)	FCLs (Neurons)	Epoch	Batch Size	Error	Accuracy (%)
10	4 (12, 10, 17, 28)	3 (65, 40, 73)	12	32	0	100
20	4 (16, 16, 28, 23)	4 (150, 10, 117, 19)	12	8	0	100
30	3 (16, 11, 22)	3 (150, 10, 78)	20	8	0.0022	99.78
40	5 (8, 16, 16, 32, 64)	3 (10, 10, 10)	20	8	0.0017	99.83
50	4 (13, 12, 24, 32)	3 (99, 109, 54)	15	8	0.0033	99.67
60	4 (8, 16, 32, 32)	3 (150, 10, 150)	12	8	0.0056	99.44
70	4 (14, 16, 25, 23)	5 (104, 150, 10, 21, 50)	17	8	0.0067	99.33
80	4 (16, 8, 32, 18)	5 (150, 128, 10, 100, 10)	15	8	0.0125	98.75
90	1 (16)	5 (105, 150, 108, 100, 47)	19	8	0.0249	97.51

The results achieved by the PSO are shown in Table 4. The results illustrate how the accuracy (best and average) decreases as the percentage of images for the testing phase increases, and this occurs because the CNN is trained with less information.

Table 4. The best, average, and worst accuracy values obtained by the PSO.

Images (Testing) %	Best %	Average %	Worst %
10	-	100	-
20	100	99.66	99.17
30	99.78	99.59	99.22
40	99.83	99.51	99.25
50	99.67	99.49	99.20
60	99.44	99.17	98.72
70	99.33	98.72	98.00
80	98.75	98.08	97.54
90	97.51	97.15	96.14

5.2. WOA Results

In Table 5, the best architectures achieved by the WOA with different percentages of images for the testing phase are shown. The best results are also obtained with 10% and 20% of images for the testing phase, where an accuracy of 100% is achieved (marked with bold text in Table 5). The best architecture can be defined as the one that uses 20% for the testing phase. This CNN model is structured as five CLs with 16, 16, 32, 32, and 64 filters, with a size of 3×3 with five FCLs with 150, 88, 150, 100, and 50 neurons and a batch size of 32 with 20 epochs.

Table 5. The best accuracy results and architectures obtained by the WOA. CLs indicates the number of convolutional layers with their number of filters, and FCLs indicates the number of fully connected layers with their number of neurons.

% Images for Testing	CLs (Filters)	FCLs (Neurons)	Epoch	Batch Size	Error	Accuracy (%)
10	3 (9, 15, 21)	4 (77, 84, 83, 27)	19	8	0	100
20	5 (16, 16, 32, 32, 64)	5 (150, 88, 150, 100, 50)	20	32	0	100
30	4 (13, 16, 32, 27)	2 (150, 143)	20	8	0.0022	99.78
40	5 (16, 13, 32, 32, 46)	4 (150, 26, 136, 56)	20	8	0.0017	99.83
50	5 (16, 13, 32, 32, 64)	5 (150, 80, 150, 100, 50)	20	8	0.0033	99.67
60	5 (16, 12, 32, 32, 64)	5 (150, 137, 53, 55, 50)	20	16	0.0056	99.44
70	4 (16, 14, 30, 32)	4 (150, 150, 114, 26)	20	8	0.0067	99.33
80	5 (16, 9, 32, 32, 54)	3 (53, 150, 150)	20	8	0.0121	98.79
90	3 (14, 10, 23)	3 (11, 96, 102)	20	8	0.0223	97.77

The results achieved by the WOA is shown in Table 6. The results show how the accuracy (best and average) also decreases as the percentage of images for the testing phase increases, except the best result using 40% of the images in the testing phase, which is superior to the best value obtained using 30%.

Table 6. The best, average, and worst accuracy values obtained by the WOA.

Images (Testing) %	Best %	Average %	Worst %
10	100	99.92	99.33
20	100	99.76	99.50
30	99.78	99.53	99.11
40	99.83	99.46	99.17
50	99.67	99.48	99.27
60	99.44	98.94	98.27
70	99.33	98.76	98.14
80	98.79	97.94	97.24
90	97.77	97.14	96.51

5.3. BA Results

The best architectures achieved by the BA with different percentages of images for the testing phase are shown in Table 7. The best result is obtained with only 10% of images for the testing phase, where an accuracy of 100% is achieved (marked with bold text in Table 7). This CNN model is structured as follows: three CLs with 11, 10, and 28 filters, with a size of 3×3, three FCLs with 121, 61, and 63 neurons, and a batch size of 16 with 14 epochs.

This architecture uses less convolutional and fully connected layers than the previous ones, which also obtained 100% accuracy.

Table 7. The best accuracy results and architectures obtained by the BA. CLs indicates the number of convolutional layers with their number of filters, and FCLs indicates the number of fully connected layers with their number of neurons.

% Images for Testing	CLs (Filters)	FCLs (Neurons)	Epoch	Batch Size	Error	Accuracy (%)
10	3 (11, 10, 28)	3 (121, 61, 63)	14	16	0	100
20	3 (14, 15, 20)	3 (66, 69, 34)	15	8	0.0017	99.83
30	4 (14, 13, 17, 31)	4 (12, 43, 10, 75)	20	8	0.0022	99.78
40	4 (15, 8, 16, 16)	3 (150, 150, 10)	20	8	0.0033	99.67
50	4 (16, 15, 32, 24)	5 (150, 150, 150, 33, 28)	20	8	0.0027	99.73
60	4 (15, 16, 26, 32)	5 (35, 150, 50, 36, 10)	20	8	0.0050	99.50
70	5 (8, 8, 32, 32, 64)	5 (42, 150, 150, 100, 50)	20	8	0.0072	99.28
80	3 (16, 8, 32)	4 (50, 29, 150, 100)	20	8	0.0109	98.91
90	2 (12, 11)	4 (71, 75, 96, 25)	12	8	0.0245	97.55

Table 8 shows the results achieved by the BA. For this metaheuristic, the accuracy (best and average) also decreases as the percentage of images for the testing phase increases, except the best result using 50% of the images in the testing phase, which is superior to the best value obtained using 40%.

Table 8. The best, average, and worst accuracy values obtained by the BA.

Images (Testing) %	Best %	Average %	Worst %
10	-	100	-
20	99.83	99.72	99.50
30	99.78	99.54	99.22
40	99.67	99.47	99.00
50	99.73	99.53	99.33
60	99.50	99.23	99.05
70	99.28	98.89	98.33
80	98.91	98.16	97.70
90	97.55	97.23	96.84

5.4. GWO Results

Table 9 shows the best architectures achieved using the GWO with different percentages of images for the testing phase. The best results are obtained with 10% and 20% of images for the testing phase as PSO and WOA, where an accuracy of 100% is achieved

(marked with bold text in Table 9). The best architecture can be defined as the one that uses 20% for the testing phase. This CNN model is structured in the following way: four convolutional layers with 10, 8, 23, and 25 filters, with a size of 3×3 with three FCLs with 42, 139, and 32 neuron and a batch size of 8 with 20 epochs.

Table 9. The best accuracy results and architectures obtained by the GWO. CLs indicates the number of convolutional layers with their number of filters, and FCLs indicates the number of fully connected layers with their number of neurons.

% Images for Testing	CLs (Filters)	FCLs (Neurons)	Epoch	Batch Size	Error	Accuracy (%)
10	4 (13, 8, 27, 24)	2 (122, 104)	17	32	0	100
20	4 (10, 8, 23, 25)	3 (42, 139, 32)	20	8	0	100
30	3 (9, 8, 22)	2 (38, 93)	10	8	0.0033	99.67
40	4 (16, 16, 16, 30)	4 (150, 67, 106, 10)	20	8	0.0025	99.75
50	5 (8, 9, 32, 19, 64)	3 (120, 81, 10)	20	8	0.0033	99.67
60	4 (9, 12, 16, 29)	5 (63, 10, 53, 15, 15)	20	8	0.0067	99.33
70	4 (8, 8, 26, 32)	3 (14, 102, 37)	16	8	0.0081	99.19
80	3 (16, 13)	1 (48)	15	8	0.0175	98.25
90	1 (15)	4 (107, 131, 117, 53)	11	8	0.0241	97.59

Table 10 shows the results obtained by the GWO. The accuracy decreases as with the other metaheuristics, but when 30% and 50% for the testing phase are used, the same result (the best value) is obtained. It is important to mention that when using 40%, the accuracy is better (the best value).

Table 10. The best, average, and worst accuracy values obtained by the GWO.

Images (Testing) %	Best %	Average %	Worst %
10	100	99.93	99.67
20	100	99.62	99.00
30	99.67	99.40	99.11
40	99.75	99.47	99.17
50	99.67	99.34	98.80
60	99.33	98.84	98.22
70	99.19	98.76	98.33
80	98.25	97.91	97.62
90	97.59	97.18	96.81

5.5. Comparison of Results

In Tables 3, 5, 7 and 9, the best architectures generated by each metaheuristic are presented, where it can be seen how the architectures can vary and still provide good results without using architectures as complex as those of the pre-trained models.

In Figure 9, the accuracy values (best, average, and worst) shown in Tables 4, 6, 8 and 10 are graphically shown. We can see that the PSO (Figure 9a) and BA (Figure 9c) always achieve an accuracy of 100% when 10% of the images are used for the testing phase (90% for training and validation). Meanwhile, the WOA and GWO only achieved the same value in some experiments using the same percentage of images. Using 50% of the percentage of images, we can see how the BA and PSO have very parallel values, which indicates that there is not much difference between their values (best, average, and worst), which could indicate greater stability between the results obtained in their experiments.

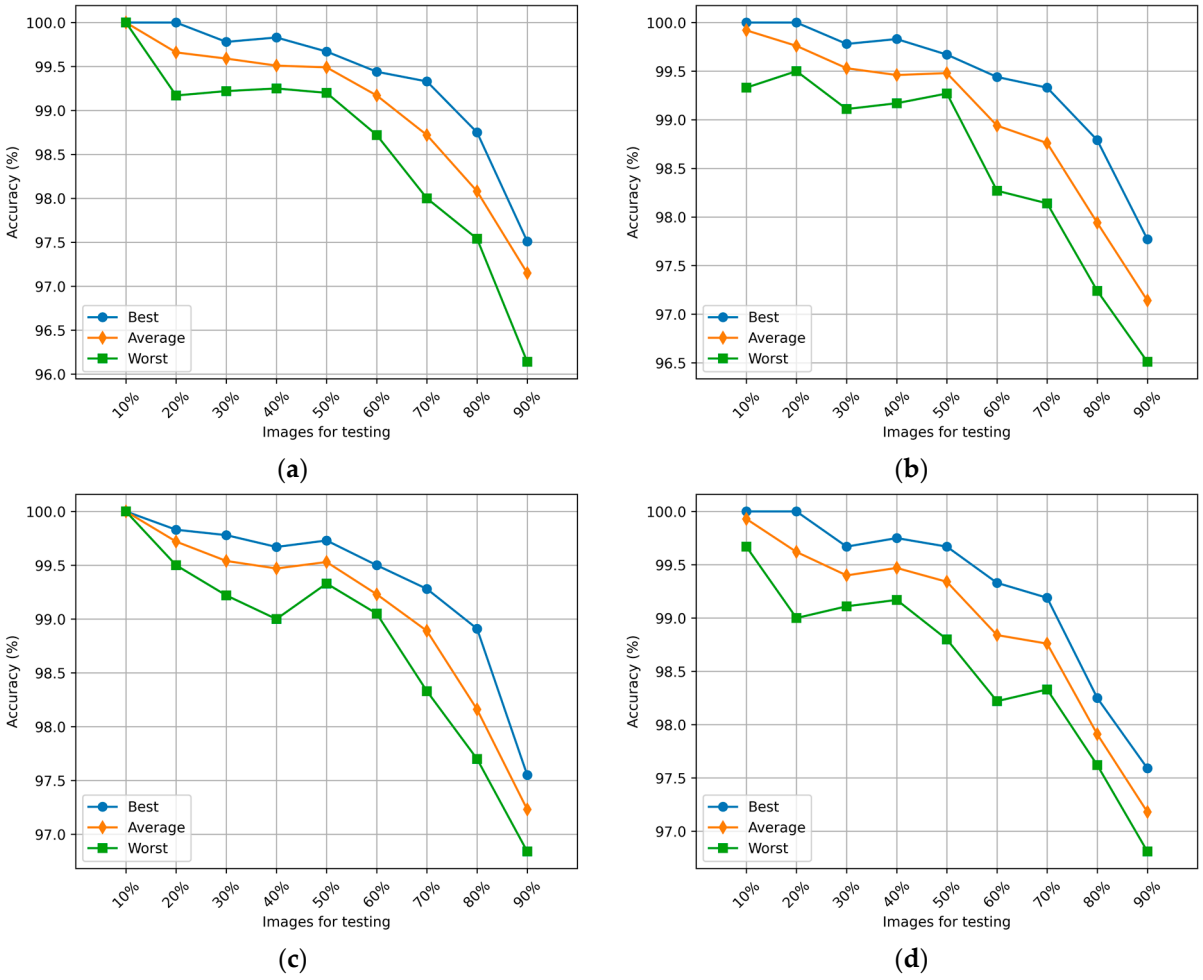

Figure 9. Accuracy values achieved by: (**a**) PSO; (**b**) WOA; (**c**) BA; (**d**) GWO using different percentages of images for the testing phase.

The average convergence during the learning phase of the 20 runs for each percentage of images (from 10 up to 90) obtained with each metaheuristic is depicted in Figure 10. It can be observed that when different percentages of images are used for the testing phase, the behavior of the PSO and BA is very similar. Even with only 10% of the images used for testing, both the PSO and BA achieve an error of 0 by iterations 6 and 4, respectively. Meanwhile, WOA and GWO exhibit similar behavior when 60%, 70%, 80%, and 90% of the images are used for the testing phase.

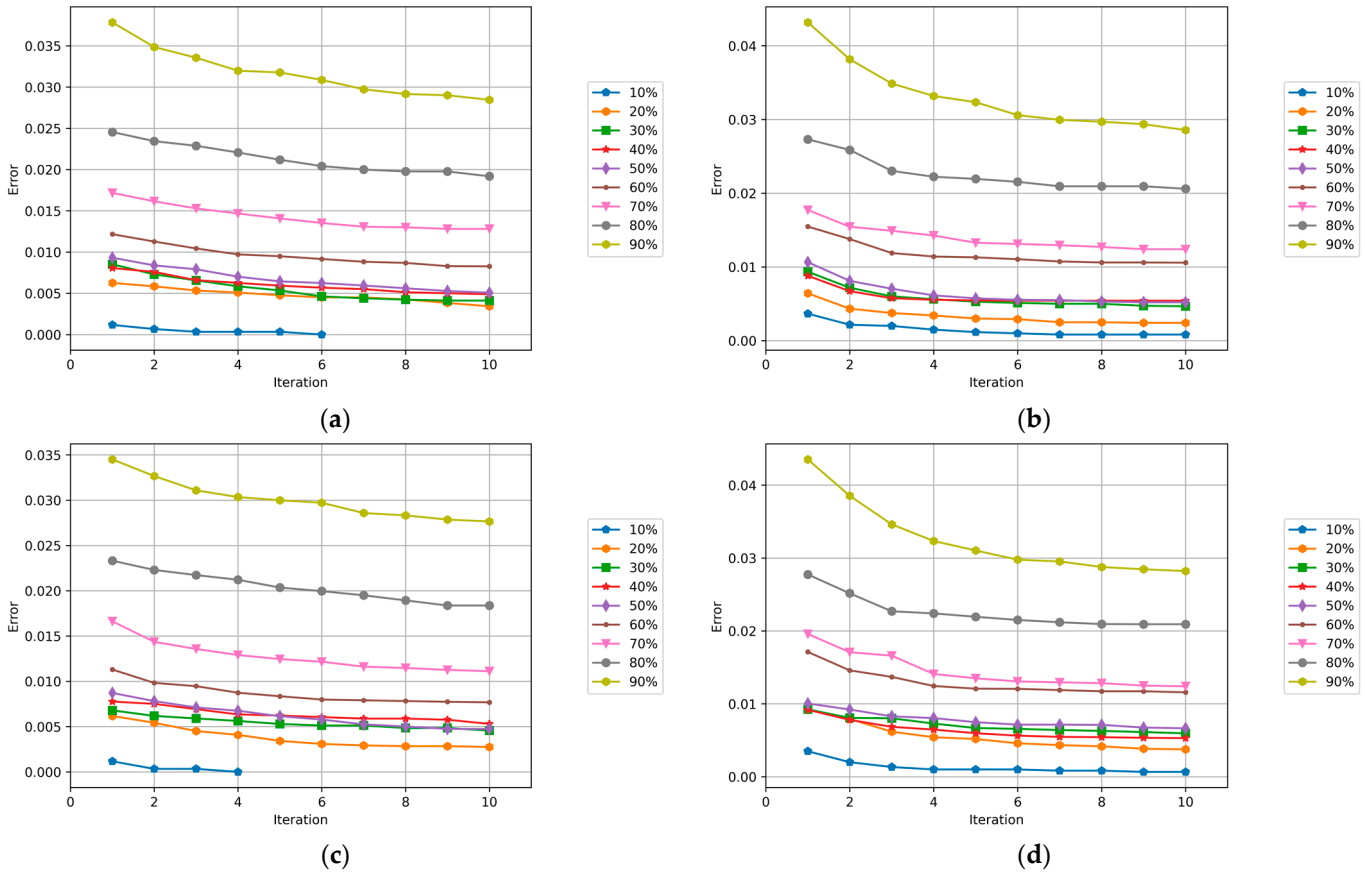

Figure 10. Convergence of accuracy error for: (**a**) PSO; (**b**) WOA; (**c**) BA; (**d**) GWO using different percentages of images for the testing phase.

The accuracy and loss curves with their respective validation of the best models are depicted in Figure 11. These models achieved an accuracy of 100%. The figure shows how the accuracy and loss have similar behavior to their validation.

Table 11 shows the averages (accuracy) obtained by each optimization algorithm. As results show, the average decreases when the percentage of images for testing increases, which means the CNN has less information to learn. Only two metaheuristics can achieve 100% accuracy: PSO and BA. Figure 12 shows graphically the accuracy achieved by the metaheuristics.

Table 11. Summary of accuracy results obtained by the metaheuristics.

Images (Testing) %	PSO %	WOA %	BA %	GWO %
10	100	99.92	100	99.93
20	99.66	99.76	99.72	99.62
30	99.59	99.53	99.54	99.40
40	99.51	99.46	99.47	99.47
50	99.49	99.48	99.53	99.34
60	99.17	98.94	99.23	98.84
70	98.72	98.76	98.89	98.76
80	98.08	97.94	98.16	97.91
90	97.15	97.14	97.23	97.18

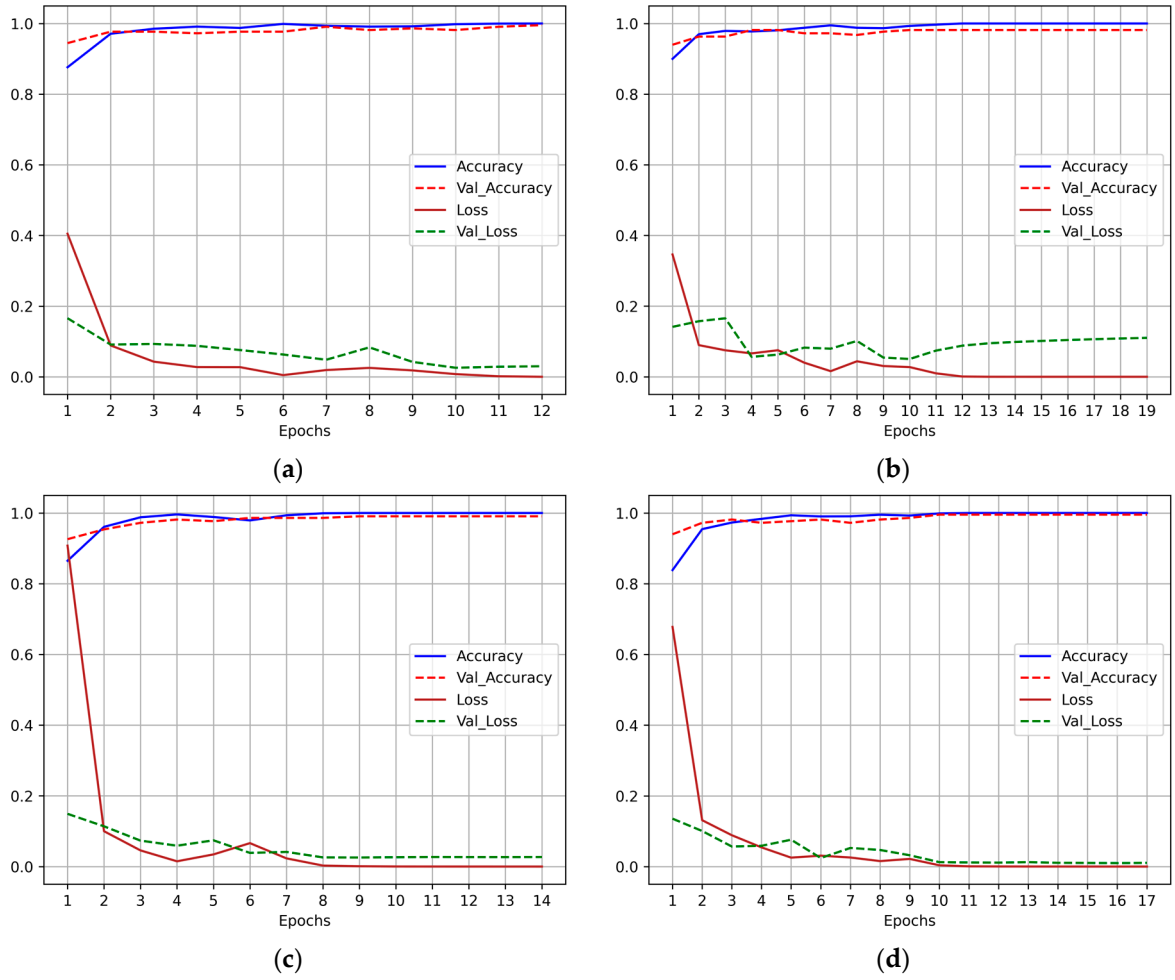

Figure 11. The accuracy and loss curves of the best models: (**a**) PSO; (**b**) WOA; (**c**) BA; (**d**) GWO.

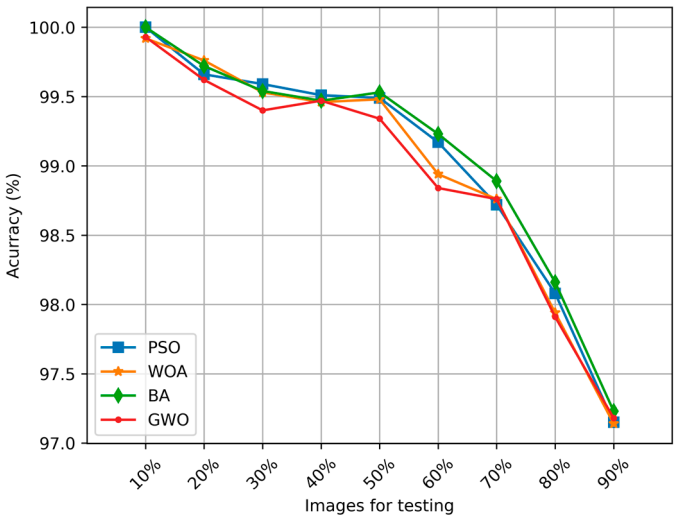

Figure 12. The average accuracy obtained by the metaheuristics.

The errors achieved by each metaheuristic are shown in Table 12. These results are utilized to perform statistical comparisons in the next section. These errors are graphically shown in Figure 13.

Table 12. Summary of error results obtained by the metaheuristics.

% Images (Testing)	PSO	WOA	BA	GWO
10	0	0.0008	0	0.0007
20	0.0034	0.0024	0.0028	0.0038
30	0.0041	0.0047	0.0046	0.006
40	0.0049	0.0054	0.0053	0.0053
50	0.0051	0.0052	0.0047	0.0066
60	0.0083	0.0106	0.0077	0.0116
70	0.0128	0.0124	0.0111	0.0124
80	0.0192	0.0206	0.0184	0.0209
90	0.0285	0.0286	0.0277	0.0282

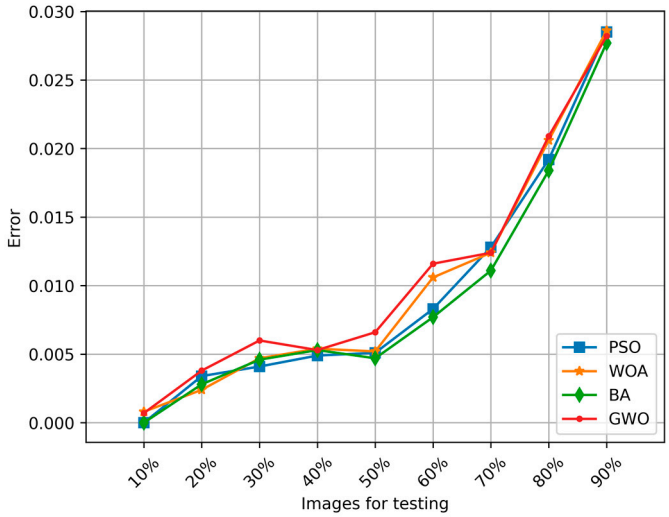

Figure 13. The accuracy error achieved by the metaheuristics.

Figures 12 and 13 graphically show the results obtained in this study. It can be seen that when a percentage between 10% and 50% is used for the testing phase, the PSO, WOA, and BA have similar good behavior. Meanwhile, when the percentage increases, it can be observed that the BA has a better accuracy, which means less error. In Table 13, the results achieved with the best average of accuracy are shown using other metrics (Recall, Precision, and F1 Score). The results show that the BA achieved better results in the other metrics, proving the effectiveness in metrics such as the F1 Score, where a combination of Recall and Precision is performed.

Table 13. Average results using Accuracy, Recall, Precision, and F1 Score.

Metric	PSO	WOA	BA	GWO
Accuracy	100	99.92	100	99.93
Recall	97.05	96.28	99.77	95.80
Precision	80.07	82.67	84.47	81.38
F1 Score	86.18	87.31	90.54	86.40

6. Statistical Comparison

This section shows statistical comparisons where the averages (errors) achieved by each optimization algorithm are used. In this work, the Wilcoxon signed-rank tests are

utilized, where the value of α depends on the statistical significance. Table 14 shows the critical values with different statistical significance levels. A significance level of 0.10 is used in this work.

Table 14. Critical values for the Wilcoxon signed-rank test.

n	α		
	0.02	**0.05**	**0.10**
9	3	6	8

Table 15 shows the results of the statistical tests performed among all the metaheuristics. The null hypothesis assumes that means are equal, which contradicts the alternative hypothesis. The null hypothesis can be rejected if the column "W" value is equal to or smaller than the "W_0" based on the critical value with a 0.10 significance level. All possible comparisons were performed among the four metaheuristics studied in this work. The results exhibit a significant difference between the PSO and GWO. Meanwhile, the BA achieves significant differences against the other metaheuristics, allowing a better face mask classification.

Table 15. Summary of Wilcoxon test results.

Methods	**Negative Sum (W−)**	**Positive Sum (W+)**	**Test Statistic (W)**	**Degrees of Freedom (m)**	**$W_0 = W\alpha,m$**
BA PSO	41	3	3	9	8
BA WOA	41	3	3	9	8
BA GWO	44	0	0	9	8
PSO WOA	34	10	10	9	8
PSO GWO	39	5	5	9	8
WOA GWO	33	10	10	9	8

The results obtained with the method applying the bat algorithm allowed us to obtain better results, especially when less percentage is utilized for the training phase of the CNNs applied to face mask classification.

7. Conclusions

In this work, four swarm intelligence metaheuristics were applied to perform a comparison. A face mask database is used as a training, validation, and testing set to prove the proposed CNN design. This database has three classes: no mask, incorrect mask, and mask. The metaheuristics applied to CNN architecture design were PSO, WOA, BA, and GWO. These algorithms were implemented to CNN optimization applied to face mask multiclass classification, where hyperparameters of CNN were sought: the number of convolutional layers, filters, number of fully connected layers, neurons, batch size, and epoch. The results showed that some average convergences of the metaheuristics have a similar behavior when different percentages of images for the testing phase are utilized. The PSO and BA achieved an average of 100% accuracy when 10% of the images for the testing phase were used (leaving 90% for training and validation), but the BA converged faster than the PSO. The Wilcoxon signed-rank tests are utilized to compare results, and there is a statistical difference when the PSO and GWO are compared. However, when comparing the BA

against PSO, WOA, and GWO, there is a statistical difference, which indicates that the BA allows for achieving better results than the other metaheuristics analyzed in this study when hyperparameters of convolutional neural networks are searched for the face mask classification. Results achieved in previous works and the results obtained in this work show that the performance of each optimization algorithm will depend on its application. In this work, only 3000 images were used, and different percentages of images were used for each phase to find optimal architectures with fewer images performing comparisons among swarm intelligence algorithms. The real implementation implies that optimized models have learned enough with the idea of not invading privacy and not having to train the models with specific persons. The optimized architectures could perform a correct face mask classification independently whether images of a person were used or not to train the model. The comparison performed in this work will allow us, as future work, to select those optimized architectures with a better percentage of accuracy and continue with the implementation in a real-time system. Although metaheuristics allow for optimal architectures with high accuracy, several limitations must be addressed in future works, such as the use of novel types of face masks not considered in this work, which would lead us to the need to evaluate their behavior. The dataset used to train and evaluate the architectures uses different face positions. However, it would be important to work with images with different kinds of illumination, especially for future work on implementing these architectures in real systems. Also, in future works, the comparison of these metaheuristics will be implemented by applying them to other intelligent techniques, such as fuzzy logic for parameter adjustment or fuzzy control.

Author Contributions: Methodology and validation, P.M.; Software, validation, writing, D.S.; Conceptualization, creation on main idea, writing—review and editing, O.C.; formal analysis, M.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank TecNM and Conacyt for their support during the realization of this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Eikenberry, S.; Mancuso, M.; Iboi, E.; Phan, T.; Eikenberry, K.; Kuang, Y.; Kostelich, E.; Gumel, A. To mask or not to mask: Modeling the potential for face mask use by the general public to curtail the COVID-19 pandemic. *Infect. Dis. Model.* **2020**, *5*, 293–308. [CrossRef] [PubMed]
2. Garcia Godoy, L.; Jones, A.; Anderson, T.; Fisher, C.; Seeley, K.; Beeson, E.; Zane, H.; Peterson, J.; Sullivan, P. Facial protection for healthcare workers during pandemics: A scoping review. *BMJ Glob. Health* **2020**, *5*, e002553. [CrossRef] [PubMed]
3. MacIntyre, C.; Cauchemez, S.; Dwyer, D.; Seale, H.; Cheung, P.; Browne, G.; Fasher, M.; Wood, J.; Gao, Z.; Booy, R.; et al. Face Mask Use and Control of Respiratory Virus Transmission in Households. *Emerg. Infect. Dis.* **2009**, *15*, 233–241. [CrossRef] [PubMed]
4. MacIntyre, C.; Chughtai, A.; Rahman, B.; Peng, Y.; Zhang, Y.; Seale, H.; Wang, X.; Wang, Q. The efficacy of medical masks and respirators against respiratory infection in healthcare workers. *Influenza Other Respir. Viruses* **2017**, *11*, 511–517. [CrossRef] [PubMed]
5. Pham-Hoang-Nam, A.; Le-Thi-Tuong, V.; Phung-Khanh, L.; Ly-Tu, N. Densely Populated Regions Face Masks Localization and Classification Using Deep Learning Models. In Proceedings of the Sixth International Conference on Research in Intelligent and Computing, Thủ Dầu Một, Vietnam, 3–4 June 2021.
6. Sethi, S.; Kathuria, M.; Kaushik, T. Face mask detection using deep learning: An approach to reduce risk of Coronavirus spread. *J. Biomed. Inform.* **2021**, *120*, 103848. [CrossRef]
7. Yu, J.; Zhang, W. Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4. *Sensors* **2021**, *21*, 3263. [CrossRef]
8. Mar-Cupido, R.; Garcia, V.; Rivera, G.; Sánchez, J. Deep transfer learning for the recognition of types of face masks as a core measure to prevent the transmission of COVID-19. *Appl. Soft Comput.* **2022**, *125*, 109207. [CrossRef]
9. Umer, M.; Sadiq, S.; Alhebshi, R.; Alsubai, S.; Hejaili, A.; Eshmawi, A.; Nappi, M.; Ashraf, I. Face mask detection using deep convolutional neural network and multi-stage image processing. *Image Vis. Comput.* **2023**, *133*, 104657. [CrossRef]

10. Ramakrishnan, K.; Balakrishnan, V.; Wong, H.; Tay, S.; Soo, K.; Kiew, W. Face Mask Wearing Classification Using Machine Learning. *Eng. Proc.* **2023**, *41*, 13.
11. Habib, S.; Alsanea, M.; Aloraini, M.; Al-Rawashdeh, H.; Islam, M.; Khan, S. An Efficient and Effective Deep Learning-Based Model for Real-Time Face Mask Detection. *Sensors* **2022**, *22*, 2602. [CrossRef]
12. Wakchaure, A.; Kanawade, P.; Jawale, M.; William, P.; Pawar, A. Face Mask Detection in Realtime Environment using Machine Learning based Google Cloud. In Proceedings of the International Conference on Applied Artificial Intelligence and Computing, Salem, India, 9–11 May 2022.
13. Mirjalili, S. *Evolutionary Algorithms and Neural Networks: Theory and Applications*, 1st ed.; Springer: London, UK, 2019.
14. Du, K.; Swamy, M. *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*, 1st ed.; Birkhäuser Cham: Berlín, Germany, 2018.
15. Hassanien, A.; Emary, E. *Swarm Intelligence: Principles, Advances and Applications*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2015.
16. Iba, H. *AI and SWARM: Evolutionary Approach to Emergent Intelligence*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2019.
17. Poma, Y.; Melin, P.; Gonzalez, C.; Martinez, G. Optimization of convolutional neural networks using the fuzzy gravitational search algorithm. *J. Autom. Mob. Robot. Intell. Syst.* **2020**, *14*, 109–120. [CrossRef]
18. Yang, X. *Nature-Inspired Computation and Swarm Intelligence: Algorithms, Theory and Applications*, 1st ed.; Academic Press: Cambridge, MA, USA, 2020.
19. Raziani, S.; Azimbagirad, M. Deep CNN hyperparameter optimization algorithms for sensor-based human activity recognition. *Neurosci. Inform.* **2022**, *2*, 100078. [CrossRef]
20. Yeh, W.; Lin, Y.; Liang, Y.; Lai, C.; Huang, C. Simplified swarm optimization for hyperparameters of convolutional. *Comput. Ind. Eng.* **2023**, *177*, 109076. [CrossRef]
21. Chawla, R.; Beram, S.; Murthy, C.; Thiruvenkadam, T.; Bhavani, N.; Saravanakumar, R.; Sathishkumar, P. Brain tumor recognition using an integrated bat algorithm with a convolutional neural network approach. *Meas. Sens.* **2022**, *24*, 100426. [CrossRef]
22. Melin, P.; Sánchez, D.; Castillo, O. Comparison of optimization algorithms based on swarm intelligence applied to convolutional neural networks for face recognition. *Int. J. Hybrid Intell. Syst.* **2022**, *18*, 161–171. [CrossRef]
23. Melin, P.; Sánchez, D.; Pulido, M.; Castillo, O. Convolutional Neural Network Design using a Particle Swarm Optimization for Face Recognition. In Proceedings of the International Conference on Hybrid Intelligent Systems, Online, 13–15 December 2021.
24. Fernandes Junior, F.; Yen, G. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm Evol. Comput.* **2019**, *49*, 62–74. [CrossRef]
25. Bashkandi, A.; Sadoughi, K.; Aflaki, F.; Alkhazaleh, H.; Mohammadi, H.; Jimenez, G. Combination of political optimizer, particle swarm optimizer, and convolutional neural network for brain tumor detection. *Biomed. Signal Process. Control* **2023**, *81*, 104434. [CrossRef]
26. Murugan, R.; Goel, T.; Mirjalili, S.; Chakrabartty, D. WOANet: Whale optimized deep neural network for the classification of COVID-19 from radiography images. *Biocybern. Biomed. Eng.* **2021**, *41*, 1702–1708. [CrossRef]
27. Knypiński, Ł. Constrained optimization of line-start PM motor based on the gray wolf optimizer. *Maint. Eng.* **2021**, *23*, 1–10. [CrossRef]
28. Nazri, E.; Murairwa, S. Classification of heuristic techniques for performance comparisons. In Proceedings of the International Conference on Mathematics, Statistics, and Their Applications, Banda Aceh, Indonesia, 4–6 October 2016.
29. Kumar, A.; Bawa, S. A comparative review of meta-heuristic approaches to optimize the SLA violation costs for dynamic execution of cloud services. *Soft Comput.* **2020**, *24*, 3909–3922. [CrossRef]
30. Fan, C.; Chung, Y. Design and Optimization of CNN Architecture to Identify the Types of Damage Imagery. *Mathematics* **2022**, *10*, 3483. [CrossRef]
31. Fregoso, J.; Gonzalez, C.; Martinez, G. Optimization of Convolutional Neural Networks Architectures Using PSO for Sign Language Recognition. *Axioms* **2021**, *10*, 139. [CrossRef]
32. Shaban Naseri, R.; Kurnaz, A.; Farhan, H. Optimized face detector-based intelligent face mask detection model in IoT using deep learning approach. *Appl. Soft Comput.* **2023**, *134*, 109933. [CrossRef]
33. Sánchez, D.; Melin, P.; Castillo, O. A Grey Wolf Optimizer for Modular Granular Neural Networks for Human Recognition. *Comput. Intell. Neurosci.* **2017**, *2017*, 4180510. [CrossRef]
34. Sánchez, D.; Melin, P.; Castillo, O. Optimization of modular granular neural networks using a firefly algorithm for human recognition. *Eng. Appl. Artif. Intell.* **2017**, *64*, 172–186. [CrossRef]
35. Campos, A.; Melin, P.; Sánchez, D. Multiclass Mask Classification with a New Convolutional Neural Model and Its Real-Time Implementation. *Life* **2023**, *13*, 368. [CrossRef]
36. Haykin, S. *Neural Networks: A Comprehensive Foundation*, 1st ed.; Macmillan: London, UK, 1994.
37. Nunes Da Silva, I.; Hernane Spatti, D.; Flauzino, A.; Bartocci Liboni, L.; Dos Reis Alves, S. *Artificial Neural Networks: A Practical Course*, 1st ed.; Springer: London, UK, 2018.
38. Aggarwal, C. *Neural Networks and Deep Learning: A Textbook*, 1st ed.; Springer: London, UK, 2018.
39. Singh, M.; Singh, G. Two phase learning technique in modular neural network for pattern classification of handwritten Hindi alphabets. *Mach. Learn. Appl.* **2021**, *6*, 100174. [CrossRef]
40. Koonce, B. *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, 1st ed.; Apress: New York, NY, USA, 2021.

41. Ozturk, S. *Convolutional Neural Networks for Medical Image Processing Applications*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2022.
42. Eberhart, R.; Kennedy, J. A New Optimizer using Particle Swarm. In Proceedings of the International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995.
43. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the IEEE International Joint Conference on Neuronal Networks, Perth, WA, Australia, 27 November–1 December 1995.
44. Eberhart, R.; Shi, Y. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, La Jolla, CA, USA, 16–19 July 2000.
45. Xin, J.; Chen, G.; Hai, Y. A Particle Swarm Optimizer with Multi-stage Linearly-Decreasing Inertia Weight. In Proceedings of the International Joint Conference on Computational Sciences and Optimization, Sanya, China, 24–26 April 2009.
46. Mirjalili, S.; Mirjalili, S.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
47. Mech, L. Alpha status, dominance, and division of labor in wolf packs. *Can. J. Zool.* **1999**, *77*, 1196–1203. [CrossRef]
48. Muro, C.; Escobedo, R.; Spector, L.; Coppinger, R. Wolf-pack (*Canis lupus*) hunting strategies emerge from simple rules in computational simulations. *Behav. Process.* **2011**, *88*, 192–197. [CrossRef]
49. Long, W.; Jiao, J.; Liang, X.; Tang, M. Inspired grey wolf optimizer for solving large-scale function optimization problems. *Appl. Math. Model.* **2018**, *60*, 112–126. [CrossRef]
50. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
51. Watkins, W.; Schevill, W. Aerial Observation of Feeding Behavior in Four Baleen Whales: Eubalaena glacialis, Balaenoptera borealis, Megaptera novaeangliae, and Balaenoptera physalus. *J. Mammal.* **1979**, *60*, 155–163. [CrossRef]
52. Yang, X. A New Metaheuristic Bat-Inspired Algorithm. In *Studies in Computational Intelligence*, 1st ed.; González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N., Eds.; Springer: London, UK, 2010; Volume 284, pp. 65–74.
53. Talbi, N. Design of Fuzzy Controller rule base using Bat Algorithm. *Energy Procedia* **2019**, *162*, 241–250. [CrossRef]
54. Yang, X. Review of meta-heuristics and generalised evolutionary walk algorithm. *Int. J. Bio-Inspired Comput.* **2011**, *3*, 77–84. [CrossRef]
55. Perez, J.; Valdez, F.; Castillo, O.; Melin, P.; Gonzalez, C.; Martinez, G. Interval type-2 fuzzy logic for dynamic parameter adaptation in the bat algorithm. *Soft Comput.* **2017**, *21*, 667–685. [CrossRef]
56. Cabani, A.; Hammoudi, K.; Benhabiles, H.; Melkemi, M. MaskedFace-Net—A dataset of correctly/incorrectly masked face images in the context of COVID-19. *Smart Health* **2020**, *19*, 100144. [CrossRef]
57. Karras, T.; Laine, S.; Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
58. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional Architecture for Fast Feature Embedding. In Proceedings of the ACM Conference on Multimedia, Orlando, FL, USA, 3–7 November 2014.
59. Campos, A.; Melin, P.; Sánchez, D. Convolutional neural networks for face detection and face mask multiclass classification. In Proceedings of the International Conference on Hybrid Intelligent Systems, Online, 13–15 December 2022.