Tuning Convolutional Neural Network Hyperparameters by Bare Bones Fireworks Algorithm

Ira TUBA[1], Mladen VEINOVIC[1], Eva TUBA[1], Romana CAPOR HROSIK[2], Milan TUBA[1]*

[1] Singidunum University, 32 Danijelova Street, 11000, Belgrade, Serbia
ituba@ieee.org, mveinovic@singidunum.ac.rs, etuba@ieee.org, tuba@ieee.org (**Corresponding author*)

[2] University of Dubrovnik, 12 Kneza Damjana Jude Street, 20000, Dubrovnik, Croatia
rcapor@unidu.hr

Abstract: Digital image classification is an important component in various applications. Lately, convolutional neural networks have been widely used as a classifier since they achieve superior results, while their application is relatively simple. In order to achieve the best possible results, tuning of the network's hyperparameters is necessary but that represents an exponentially hard optimization problem with computationally very expensive fitness function. The swarm intelligence algorithms have been proven to be effective in solving such exponentially hard optimization problems, however their application to this particular problem has not been sufficiently studied. In this paper, convolutional neural network hyperparameters were tuned by the bare bones fireworks algorithm. The quality of the proposed method was tested on two standard benchmark datasets, CIFAR-10 and MNIST. The results were compared to CIFAR-Net, LeNet-5 and the networks optimized by the harmony search algorithm and the proposed method achieved better results considering the classification accuracy. The proposed method for CNN hyperparameter tuning improved the classification accuracy up to 99.34% on the MNIST dataset and up to 75.51% on the CIFAR-10 dataset compared to 99.25% and 74.76% reported by another method from the specialized literature.

Keywords: Convolutional neural networks, Hyperparameters tuning, Optimization, Swarm intelligence, Bare bones fireworks algorithm.

1. Introduction

In recent history, several qualitative technological advancements have made a significant impact on everyday life. Most of them were in the field of computer science and technology. PC, the Internet, and smartphones are some examples. The latest revolutionary progress has been made by the convolutional neural networks (CNN). Convolutional neural networks are a specific type of deep neural networks that use kernels to process neighboring inputs so that their spatial (or temporal) correlation is preserved. As such, CNN are suitable for the classification of different signals, predominantly digital images. The accuracy of digital image classification has been significantly improved by the CNN, even without any feature extraction or preprocessing. This facilitated revolutionary advancements since digital image classification is a crucial element in numerous applications, for example, medical image analysis, autonomous vehicles, face recognition, etc.

Even though convolutional neural networks only recently have been widely applied and researched, the concept of the CNN was introduced several decades ago. One of the first applications of the CNN was introduced by LeCun et al. (1989) for handwritten digit recognition. The proposed CNN was based on the work of Fukushima (1980) where a network that recognizes complex patterns based on the low-level complex patterns was presented. The recognition results were significantly improved by the CNN, but due to limited computational power and lack of large datasets that could be used for training there has not been any substantial further research and applications of the CNN. However, nowadays it is possible to train a deep neural network on an ordinary computer. As a result, the CNN have been widely studied and used in numerous practical applications and they achieved revolutionary results. The CNN were used in applications for solving problems in healthcare (Anwar et al., 2018; Pereira et al., 2016; Shen et al., 2017), security (Zheng et al., 2017), agriculture (Boulent et al., 2019), autonomous vehicles (Kebria et al., 2019), astronomy (Jia, Liu & Sun, 2020), and many other areas (Rawat & Wang, 2017).

Nowadays, powerful hardware and available software tools make it relatively easy to create a CNN that achieves significantly higher classification accuracy compared to other classification methods. However, creating the optimal CNN is a challenging problem since a CNN contains numerous hyperparameters that need to be properly selected. CNN parameters are adjusted during the training process while

https://doi.org/10.24846/v31i1y202203

hyperparameters are parameters that control that process. The number and type of layers, activation function, optimization algorithm are some examples of the CNN hyperparameters. Even if these hyperparameters are not set to the optimal values, most likely the results will be better than those obtained by using other classification methods. To improve the accuracy of the CNN classification, hyperparameters should be fine-tuned specifically for each problem. Since the number of hyperparameters and the range of possible values for each of them are huge, there is no deterministic method for solving this problem in a reasonable time. In practice, in many cases, CNN hyperparameter tuning is done by guessing and estimating. Researchers select values for the hyperparameters that they believe would produce good results based on the experience and the knowledge about the considered problem. Since this method has been often used, it was named. Finding a more appropriate method is an emerging research topic. Since adjusting CNN hyperparameters is an exponentially hard optimization problem, some metaheuristics that proved to be efficient for solving this class of problems could be used. One class of algorithms successfully applied to different hard optimization problems is the class of nature inspired, especially swarm intelligence (SI) algorithms (Liu et al., 2021; Al Harthi et al., 2021). There are some initial studies on the application of the SI algorithms on tuning CNN hyperparameters, but it still represents a new and insufficiently studied research topic. Usually, based on the nature of the CNN hyperparameters tuning problem, it is not possible to apply SI algorithms directly. It is necessary to adjust them for integer search space and for providing a good solution with a small number of fitness function evaluations.

In this paper, a recent SI algorithm was adapted, namely the bare bones fireworks algorithm (BBFWA) (Li & Tan, 2018), for tuning some of the CNN hyperparameters. The network architecture, which includes hyperparameters, was established based on the results known in the literature (Lee, Park & Sim, 2018), and the BBFWA was used for finding the optimal kernel size, the number of feature map channels, padding and stride in convolutional layers, and padding and stride in max-pooling layers. The BBFWA was adjusted for the considered problem by selecting the appropriate fitness function, setting the parameters, and adjusting it for integer search space. The classification accuracy was used as fitness function. The proposed method was compared with standard LeNet-5 and CifarNet, as well as with another SI approach from the literature, the modified harmony search algorithm (Lee, Park & Sim, 2018).

The rest of the paper is organized as follows. Related work is presented in Section 2. Convolutional neural networks and swarm intelligence algorithms are explained in Section 3. The bare bones fireworks algorithm adjusted for CNN hyperparameters optimization is presented in Section 4. Section 5 contains a comparison of the proposed method with methods from the literature and an analysis of the obtained results. Finally, Section 6 concludes the paper and proposes future research.

2. Related Work

In recent years, CNNs have been intensively studied. They were studied theoretically and tested on benchmark datasets, but also applied to various real-life problems. Currently, the emerging research topic is CNN hyperparameters tuning. As mentioned before, one promising approach is the application of different optimization metaheuristics, especially evolutionary and swarm intelligence algorithms. The application of the SI algorithm to the CNN hyperparameters tuning problem is a relatively new research topic and there is a relatively small number of papers in the specialized literature that tackle this problem, but the results are very promising.

There are several modifications and adaptations of one of the earliest swarm intelligence algorithms, particle swarm optimization (PSO), for CNN hyperparameter tuning. Sinha, Haidar & Verma (2018) proposed the PSO for simple CNN optimization. The PSO was used to set hyperparameters and choose between two possible CNN architectures. The quality of the proposed method was tested on two benchmark datasets, CIFAR-10 and Road Side Vegetation Dataset. The proposed method achieved better results compared to the AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) and grid search method.

Canonical PSO with three additional mechanisms for exploration and exploitation improvements and faster fitness function evaluation was proposed by Wang, Zhang & Zhang (2019). The improved PSO was compared with four other versions of the PSO and tested on the standard CIFAR-10 benchmark dataset. Hyperparameters were determined for the AlexNet architecture. The improved PSO was used to determine the kernel size, the number of kernels, stride and padding of each layer. In comparison to other PSO versions as well as to other approaches from the literature, the proposed modified PSO achieved higher classification accuracy.

Another adaptation of the PSO for finding the optimal CNN architecture named quantum behaved Particle Swarm Optimization with binary encoding (BQPSO) was proposed by Li et al. (2019). The BQPSO was used to determine kernel size, the number of feature maps, stride and padding of the convolutional layers, and padding, kernel size, and type of pooling layers. The number of neurons in fully connected layers, as well as the number and order of different layers, were also set by the BQPSO. The proposed method was tested on the complex MNIST dataset, MNIST with rotated digits plus background images (MRDBI). The obtained results were comparable to those obtained by other approaches while the main contribution was the automatic architecture design of the CNN.

Singh, Chaudhury & Panigrahi (2021) proposed a multilevel PSO for finding optimal CNN architecture and its hyperparameters. The proposed method was tested on 5 benchmark datasets, MNIST, CIFAR-10, CIFAR-100, Convex Sets, and MDRBI, and achieved the classification accuracy of 99.13%, 87.34%, 66.97%, 90.33% and 65.77%, respectively.

The PSO approach for CNN hyperparameter tuning was also proposed by Gao et al. (2020). The binary coding was used for describing the CNN architecture and hyperparameters such as kernel size, number of kernels, type of pooling layer, size of pooling. For finding the optimal solution, gradient-priority particle swarm optimization was proposed. The method was tested on EEG signals and used for emotion recognition. In comparison to four other methods, the proposed method achieved higher classification accuracy.

Another swarm intelligence algorithm, artificial bee colony optimization (ABC), was used for finding optimal hyperparameters of the CNN (Zhu et al., 2019). The ABC algorithm was used to determine the optimal values for 13 hyperparameters including the number of each type of layers, kernel sizes, learning rate, batch size and dropout probability. The proposed method was tested on the MNIST dataset and the results were compared with those obtained by three other approaches from the specialized literature. The ABC method obtained a lower classification error compared to the other approaches.

Shukla, Koley & Ghosh (2020) presented an optimized CNN for transmission line protection. The CNN was optimized by the grey wolf optimization algorithm. Classification accuracy was used as the fitness function for determining the number of convolutional layers, kernel size of convolutional and pooling layers, and dropout probability. The proposed method was tested on various scenarios and obtained satisfying results.

The harmony search algorithm (HS) was used for tuning hyperparameters of CNNs by Rosa et al. (2015). They presented the results obtained by the CNNs optimized by the original harmony search algorithm as well as by three modified versions of the HS. The proposed methods were tested on CIFAR-10 and MNIST datasets and the results were compared with those achieved through random search and with parameters from the Caffe library (Jia et al., 2014). The modified HS versions outperformed other methods while among them the best one was self-adaptive global best harmony search where a mechanism for dynamic HS parameters tuning was presented.

A similar idea of the harmony search algorithm was proposed by Lee, Park & Sim (2018). They proposed the parameter-setting-free harmony search (PFS-HS) for setting CNNs hyperparameters. More details about this method are given later since it was used for comparison with the method proposed in this research work.

There were also other swarm intelligence algorithms proposed for optimizing CNN such as the firefly algorithm (Bezdan et al., 2020; Strumberger et al., 2019b), monarch butterfly optimization algorithm (Bacanin et al., 2020), tree growth algorithm (Strumberger et al., 2019a).

3. Convolutional Neural Networks and Swarm Intelligence Algorithms

Convolutional neural networks have been widely used for solving different classifications of digital images and sound signals since numerous software frameworks made it relatively easy to implement and train the CNN. Even with the default hyperparameters values from the used library, it is possible to make a CNN that outperforms other machine learning methods. Besides the good results, another advantage is that usually there is no need for image pre-processing or feature extraction but the raw data can be used as the input. In most cases, a simple resizing of images is the whole required pre-processing.

Even though the CNNs achieve excellent results and are relatively simple to use, there are some challenges. One of the main challenges is hyperparameter tuning. CNNs have a large number of hyperparameters that should be adjusted individually for each problem.

A convolutional neural network represents a special type of artificial neural networks that uses convolution operation instead of matrix multiplication. The architecture of one CNN consists of the input and the output layers and the hidden layers between them, same as in the artificial neural networks (ANN). The difference is that hidden layers in a CNN can be fully connected, convolutional, or pooling layers. Fully-connected layers are equivalent to the hidden layers in an ANN and one or more of them are placed before the output layer. In the convolutional layer, a kernel which is a weighted matrix is used to convolve the input and produce the output. A CNN can have one or more convolutional layers. Parameters of the convolutional layer include the size of the kernel and the number of input and output feature map channels along with the stride and padding. In the first convolutional layer, the number of input feature map channels is equal to the depth of the image and in the following layers, the number of output feature map channels is equal to the number of input channels of the next layer. The convolutional layer can be followed by a pooling layer that reduces the dimension of the data. Parameters of the pooling layer include stride, padding, kernel size, type of pooling while the number of feature map channels remains the same but the dimension of each feature map is decreased. There are two most commonly used pooling types, max and average. Besides the hyperparameters that were mentioned, some hyperparameters are also part of ANN such as learning rate, an optimization method for backpropagation, loss function, number of epochs, batch size, the dropout rate for regularization. Also, the number of convolutional, pooling, fully connected layers and their order should be determined.

Finding the optimal values for all hyperparameters is a hard optimization problem. Additionally, the calculation of the fitness function requires the training of a complete CNN, so it represents a rather time-consuming operation which consequently limits the number of fitness function evaluations.

Same as for other machine learning algorithms, hyperparameters tuning can be done by simple random search or grid search. The problem with these methods is that only part of the search space can be covered and the results depend on the chosen grid values.

A better approach is to use an optimization metaheuristic that has been proved to be efficient for tackling similar kinds of problems. One approach is to use swarm intelligence (SI) algorithms. The main idea of SI algorithms is to use a population of simple agents that exchange information between themselves and through iterations, they improve the quality of the best found solution. A simple agent that is moved around by a specific rule represents one solution in the search space. In each SI algorithm, there are two main mechanisms or rules of generating new candidate solutions based on the previous ones, named exploration and exploitation. Exploration is used to ensure the global search, to explore the whole search space to find promising areas where the optimal solution could be. After finding these areas, it is necessary to exploit the quality of solutions in these particular areas. For that, the exploitation operator is used. Defining proper exploration and exploitation operators, as well as the balance between them, are the main parts of each SI algorithm.

In most of the cases, the application of the SI algorithms for CNN hyperparameters tuning, is not straightforward and some adaptations are needed. Common adaptation is related to the fact that the SI algorithms were originally

made for real search space while some CNN hyperparameters are integers. Depending on the chosen subset of hyperparameters that should be tuned, the SI algorithms may need an adjustment for integer solutions. An adjustment for the integer search space can be simple if the integers are from a large interval. In that case, the SI algorithm can be used as before with the addition of rounding the obtained real values to the nearest integer. The problem is if the integers are from the small interval, i.e., there are a small number of possible values for a certain variable. In that case, a slight change in the solution can lead to a drastic change in fitness function value and by that the whole search mechanism in the SI algorithm is compromised. Thus, changes in solution representation and/or fitness function should be introduced.

Another adaptation that is usually necessary is related to the fact that the fitness function calculation is time-consuming. Training a convolutional neural network takes much more time than most fitness functions, so the fitness function evaluation number is significantly reduced. In order to find a solution in such a small number of iterations, sometimes it is necessary to add constraints or to improve and modify the exploration and exploitation operators of the SI algorithm.

4. Adjusted Bare Bones Fireworks Algorithm for CNN Hyper-Parameters Tuning

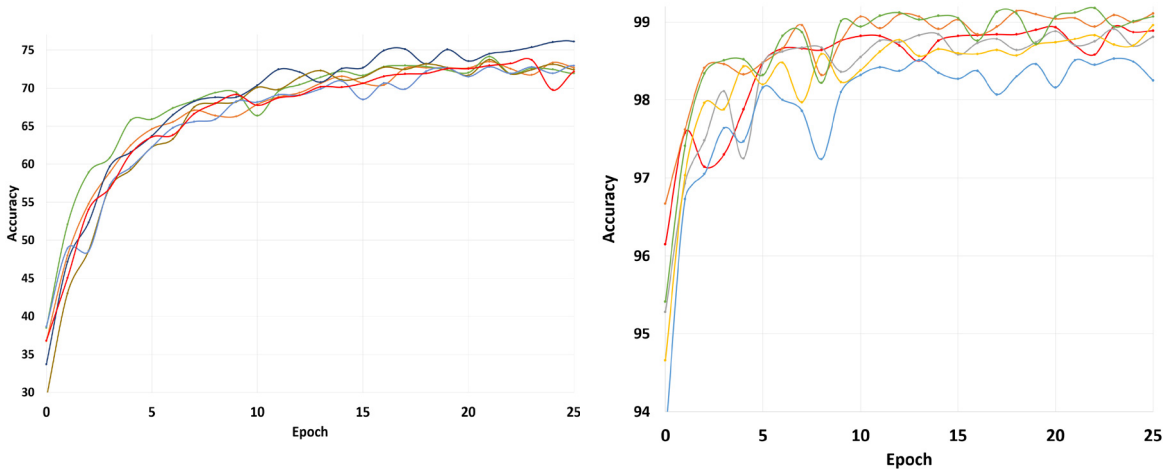
The fireworks algorithm was initially proposed by Tan and Zhu (2010). The algorithm was widely used which resulted in numerous modifications and enhancements. The improved versions were presented periodically and one of the recent versions is the bare bones fireworks algorithm (BBFWA). The BBFWA was proposed by Li and Tan (2018).

Unlike other FWA versions, the BBFWA, has a fixed number of solutions that are generated at each iteration. At each iteration, n solutions are generated around the current best solution. Among all the solutions in one iteration, the best one is saved for the next iteration. Exploration and exploitation are controlled by the size of the space around the current best solution where the new solutions will be generated. If the best solution is found among the newly generated solutions, i.e., the best solution has changed, it can be considered that the search space is still not well explored thus the size of the space around the best solution where the new solutions will be generated is increased. However, if the best solution remains the same then the potentially good area of the search space is found and it should be exploited carefully. This is implemented by narrowing the search space around the best solution.

To implement the BBFWA, two parameters were introduced, C_a and C_r, the factor of increasing and the factor of decreasing the search space around the current best solution, respectively. In the beginning, the whole search space should be covered so the size of the space around the best solution, A, is set to $A=Ub-Lb$, where U_b and L_b are the upper and the lower bounds of the search space, respectively. The other parameter of the BBFWA is the number of solutions (n) that are generated in each iteration around the current best solutions. The stopping criterium is usually the maximal number of iterations, the number of fitness function evaluations, the number of successive iterations where the best solution remains the same, etc.

The quality of the solution is defined by the fitness function. For the CNN hyperparameter tuning, commonly loss function or classification accuracy are used as the fitness function. In literature, some papers such the research work of Lee, Park and Sim (2018), used the value of the loss function after one epoch as the fitness function. To define the fitness function, an experiment was conducted in the present work to check if one epoch is enough to determine the quality of the solution. Random solutions were generated and the CNN was trained through 25 epochs. The classification accuracy was checked after each epoch. The graph that represents the classification accuracy after each epoch is presented in Figure 1. The accuracy on the CIFAR-10 dataset is presented in Figure 1a and the results on the MNIST dataset are in Figure 1b.

Based on the results, it was determined that the quality of the solution cannot be defined by the classification accuracy or the loss function after just one epoch since there are a lot of oscillations in the later epochs. After 10 or more epochs, solutions that were among the worst ones at the beginning could end up as the best ones.



(a) (b)

Figure 1. Selecting time-efficient fitness function based on the classification accuracy of the BBFWA candidate solution through 25 epochs on a) CIFAR-10 and b) MNIST datasets

Considering this, classification accuracy was used after the number of epochs as the fitness function. From the graph presented in Figure 1, it can be empirically concluded that after the first 20 epochs the best solutions are stabilized and discovered. Thus, to minimize the fitness function evaluation time, while having a proper solution rating, the epoch number in the fitness function was set to 20. In this paper, for the fitness function the classification accuracy was used instead of the loss function.

Algorithm 1. Pseudo-code of the proposed method

Initialization
Randomly generate one candidate solution x, a vector of hyperparameter values.
Round the solution values to the nearest integers and evaluate that solution.
Save solution x as the current best solution.
$A=U_b-L_b$
repeat
Generate n candidate solutions (s_i, $i=1, 2, ..., n$) within the range $[x-A, x+A]$.
Round all solutions to the nearest integers.
Evaluate the generated solutions s_i, $i=1, 2, ..., n$, i.e., train the CNN with the generated hyperparameter values and return the classification accuracy as the quality of the solution.
if $\min(f(s_i))<f(x)$
$x = s_i$
$A = C_a * A$
else
$A = C_r * A$
end if
until stopping criteria is reached.
return candidate solution x
Train the CNN with the hyperparameter values as in a candidate solution x and test the CNN on the test set
return classification accuracy on the test set

5. Experimental Results

The proposed BBFWA method for CNN hyperparameter optimization was implemented in Python 3.8 where the PyTorch package was used for the CNN implementation. The experiments were done on the platform with Intel © Core™ i7-11700K CPU at 5GHz, 16GB RAM, NVIDIA RTX 2060 graphic card, and Windows 11 Professional OS.

The proposed method was tested on two standard benchmark datasets, MNIST (LeCun, Cortes & Burges, 1998) and CIFAR-10 (Krizhevsky & Hinton, 2009). In order to test the quality of the proposed method, it was compared with the modified harmony search algorithm (HS) named parameter-setting-free HS (PSF-HS) (Lee, Park & Sim, 2018) and two well-known and commonly used CNN for comparison, LeNet-5 for the MNIST dataset and CifarNet for the CIFAR-10 dataset. The HS parameters that control exploration and exploitation were set automatically based on the quality of the results obtained in each iteration. The PFS-HS algorithm was used for finding optimal kernel size, the number of channels, padding and stride in each layer.

5.1 Experimental Setup

The MNIST dataset (LeCun, Cortes & Burges, 1998) is a widely used benchmark dataset for image classification. The dataset contains images of handwritten digits of size 28x28x1. In total,

there are 70,000 images of 10 classes where 60,000 are in the training set, while the remaining 10,000 are used for testing.

In the paper used for comparison (Lee, Park & Sim, 2018), the architecture of the convolutional neural network for the MNIST dataset was based on the LeNet-5 network (LeCun et al., 1998).

This network contains two convolutional layers, each one followed by a pooling layer. At the end, there are two fully connected layers and the output layer. The number of neurons in fully connected layers are 120 and 84, respectively, and 10 neurons, one for each class, are in the output layer. Since the input size of the LeNet-5 network is 32x32x1, the original images from the MNIST were resized by adding zero paddings which was also done in (Lee, Park & Sim, 2018).

Lee, Park & Sim (2018) compared their PSF-HS approach with the original LeNet-5 network. To ensure a fair comparison, they included constraints on the complexity of the network. The number of weights and biases that should be learned in the network was constrained by the number of learnable parameters in the LeNet-5. This number can be calculated based on the number of weights in each convolutional and fully connected layer as:

$$conv\_param = \left(\left(k_x \cdot k_y \cdot d\right)+1\right)\cdot c \quad (1)$$

$$FC\_param = fc\_c \cdot \left(fc\_p+1\right) \quad (2)$$

where *conv_param* is the number of learnable parameters in the convolutional layer with the kernel size of $k_x x k_y$, d input channels, and *c* output channels. The *FC_param* is the number of weights in the fully connected layer with *fc_c* output channels and *fc_p* input channels. The total number of learnable channels is:

$$learnable\_param = \sum conv\_param + \sum FC\_param \quad (3)$$

The LeNet-5 has 61,706 learnable parameters.

The second dataset was the CIFAR-10 (Krizhevsky & Hinton, 2009), another widely used benchmark dataset for image classification. It contains 60,000 color images of size 32x32x3. Images were divided into training and testing set with 50,000 and 10,000 samples, respectively. Each class contains the same number of images, i.e., 5000 images of each class in the training set and 1000 images of each class in the testing set. CIFAR-10 consists of images of the airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, sheep, and trucks.

The architecture of the CNN used for the classification of the CIFAR-10 dataset was based on the CifarNet (Krizhevsky & Hinton, 2009).

The CifarNet has three convolutional layers, each of them followed by a pooling layer. In the end, there is a fully connected layer with 64 neurons and the output layer with a neuron for each class.

The PSF-HS algorithm (Lee, Park & Sim, 2018) was used for finding the values of each of the following CNN hyperparameters: for each convolutional layer in the network, kernel size, number of feature map channels, padding, and stride, for the pooling layer, the padding and stride. The dimension of the optimization problem was:

$$d = 4NoConv + 2NoPool \quad (4)$$

where *NoConv* is the number of convolutional layers in the CNN while the *NoPool* is the number of pooling layers. To fairly compare the proposed method with the PSF-HS algorithm, the same constraints for the kernel sizes were set: the size of the convolutional kernel should be an odd positive integer while the size of the pooling kernel was larger or equal to 2. The loss function was a cross-entropy loss.

The other hyperparameters of the CNN were not reported in (Lee, Park & Sim, 2018) thus, they were set as follows: the pooling layers were max pooling, the optimization algorithm was set to be a stochastic gradient descent (SGD), activation function was ReLU.

In this paper, the learning rate was set empirically. It has been observed that the model tends to start overfitting, i.e., the loss function is decreasing along with the classification accuracy obtained on the testing set. Based on the initial experiments, the learning rate was initially set to 0.003 and gradually decreased to 0.0005 based on the epoch number. The rest of the hyperparameters were not explicitly set but the default values from the PyTorch package have been used.

The parameters of the BBFWA, C_a and C_r, were set empirically. The tests on both datasets were run with different values for C_a and C_r, and it was established that the best results were obtained when C_a was 0.5 and C_r was 1.3. For the MNIST dataset, the number of fitness function evaluations was set to 41, where 5 new solutions were generated in each of 8 generations. For the CIFAR-10 dataset, the number of fitness function evaluations was set to 22 and 3 new solutions were generated in each of 7 generations.

5.2 BBFWA-CNN for the MNIST

Lee, Park & Sim (2018) did not specify the upper bound of the search space so in this paper the upper bounds for each considered hyperparameter were set empirically. The lower and the upper bounds for hyperparameters of the CNN for the MNIST dataset were set as presented in Table 1.

Table 1. The lower and the upper bounds for hyperparameters of the MNIST dataset

Layer	Parameter	Lb	Ub
Convolutional layer 1	Kernel size (k1)	5	13
	Number of feature maps (c1)	4	14
	Padding (p1)	2	8
Pooling layer 1	Kernel size (k2)	2	8
Convolutional layer 2	Kernel size (k3)	11	25
	Number of feature maps (c2)	5	12
	Padding (p2)	0	4
Pooling layer 2	Kernel size (k4)	4	10

In this paper, solutions generated by the BBFWA that build a more complex network than LeNet-5 (with more learnable parameters) were rejected and replaced by a new one.

To reduce the training time, it was empirically established that if the solution achieves an accuracy which is lower than 98.8% after 5 epochs, the solution will not achieve the desired accuracy so training is stopped and the accuracy after 5 epochs is returned as the fitness function value. For other solutions, the network was trained for 20 epochs. Most of the time in one run of the BBFWA is spent for fitness function evaluation, hence reducing time for one evaluation is important. After the BBFWA finishes, the CNN was trained with the obtained hyperparameter values through 40 epochs as in (Lee, Park & Sim, 2018).

It should be mentioned that the PSF-HS proposed in (Lee, Park & Sim, 2018) involved approximately 19,000 iterations with a population size of 10, which means 190,000 fitness function evaluations, which is significantly more efficient when compared to the present approach that took only 40 fitness function evaluations. Since the proposed fitness function evaluation includes 20 epochs, it is equivalent to 800 fitness function evaluations of the PSF-HS method. The stopping criterium for the PSF-HS method was a convergence of all the ten solutions into one, which is why there were so many iterations. The classification accuracy through iterations was not reported, so it is not known if the reported result was achieved in the earlier stages of the algorithm.

Table 2 illustrates the classification accuracy obtained by the original LeNet-5 network, the PSF-HS method proposed by Lee, Park & Sim (2018) and the BBFWA method proposed in this paper. The proposed BBFWA method achieved the highest accuracy compared to the LeNet-5 and PSF-HS methods. The optimal hyperparameters that were used are presented in Table 3.

Table 2. Classification accuracy on the test set of the MNIST data

Method	Accuracy (%)
LeNet-5	98.94
PSF-HS	99.25
BBFWA	**99.34**

Table 3. CNN hyperparameters of the MNIST dataset

Layer	Hyperparameters
Conv1	k1=13, c1=13, s1=1, p1=5
Pooling 1	k2=7, s2=1
Conv2	k3=17, c2=12, s3=1, p2=4
Pooling 2	k4=9, s4=4

The total number of learnable parameters (weights and biases) was 60,308, while the CNN obtained by PSF-HS had 60,552 and LeNet-5 had 61,706. This limitation of the number of learnable parameters is rather artificial and it had been included for ensuring a fair comparison, but

removing the limitation could produce a CNN with higher classification accuracy. The average execution time for one run was 3332.61 seconds.

5.3 BBFWA-CNN for the CIFAR-10 Data Set

The CNN's architecture for the CIFAR-10 dataset was based on the CifarNet (Krizhevsky & Hinton, 2009). The lower and the upper bounds for each of the considered hyperparameters are presented in Table 4.

Table 4. The lower and the upper bounds for hyper-parameters of the CIFAR-10 dataset

Layer	Parameter	Lb	Ub
Convolutional layer 1	Kernel size (k1)	3	10
	Number of feature maps (c1)	16	51
	Padding (p1)	1	4
Pooling layer 1	Kernel size (k2)	2	5
Convolutional layer 2	Kernel size (k3)	3	10
	Number of feature maps (c2)	16	51
	Padding (p2)	1	4
Pooling layer 2	Kernel size (k4)	2	5
Convolutional layer 3	Kernel size (k5)	5	12
	Number of feature maps (c3)	10	31
	Padding (p3)	1	4
Pooling layer 3	Kernel size (k6)	2	5

The limitation for the number of learnable parameters (weights and biases) was based on the CifarNet and it is 120,042. When the generated model contained more parameters, it was rejected and a new model was proposed.

To reduce the execution time of one run, if the accuracy after 5 epochs is not greater than 70%, the training process is stopped and, so far, the found accuracy was returned as the fitness function value. The PSF-HS method proposed in (Lee, Park & Sim, 2018) found the CNN hyperparameters after approximately 16,000 iterations. Since the population size of the PFS-HS was 20, this involved 320,000 fitness function evaluations or CNN epochs. The proposed BBFWA method has 22 fitness function evaluations and, in each evaluation, there is a maximal 20 epoch, which represents a total of 440 epochs, a number significantly lower than 320,000 used in (Lee, Park & Sim, 2018).

The results obtained by the original CifarNet (Krizhevsky & Hinton, 2009), the PSF-HS (Lee, Park & Sim, 2018) and the proposed BBFWA are presented in Table 5. The obtained values of hyperparameters are presented in Table 6.

Table 5. Classification accuracy of the CIFAR-10 test dataset

Method	Accuracy (%)
CifarNet	73.32
PSF-HS	74.76
BBFWA	**75.51**

Table 6. CNN hyperparameters of CIFAR-10

Layer	Hyperparameters
Conv1	k1=9, c1=42, s1=1, p1=2
Pooling 1	k2=3, s2=2
Conv2	k3=5, c2, s3=1, p2=2
Pooling 2	k4=4, s4=1
Conv3	k5=11, c3=24, s5=1, p3=2
Pooling 3	k6=3, s6=1

The number of CNN learnable parameters is 107,322 which is smaller compared to both, CifarNet (120,042) and PSF-HA CNN (114,794). The average execution time for one run was 3515.42 seconds.

6. Conclusion

The work of this paper proposes an adjusted bare bones firework algorithm for finding a selected subset of CNN's hyperparameters and tests it on the MNIST and CIFAR-10 datasets. In comparison to the HS approach from the literature (Lee, Park & Sim, 2018), the proposed method increased the classification accuracy for both datasets, while reducing the network complexity. The proposed BBFWA approach needed significantly fewer fitness function evaluations. The classification accuracy was increased from 99.25% to 99.34% on the MNIST data and from 74.76% to 75.51% on the CIFAR-10. Future work will include more hyperparameters, other optimization algorithms, their modification and hybridization, as well as testing on different datasets.

REFERENCES

Al Harthi, M., Ghoneim, S., Elsayed, A., El-Sehiemy, R., Shaheen, A. & Ginidi, A. (2021). A Multi-Objective Marine Predator Optimizer for Optimal Techno-Economic Operation of AC/DC Grids, *Studies in Informatics and Control*, 30*(2)*, 89-99. DOI: 10.24846/v30i2y202108

Anwar, S. M., Majid, M., Qayyum, A., Awais, M., Alnowami, M. & Khan, M. K. (2018). Medical Image Analysis Using Convolutional Neural Networks: A Review, *Journal of Medical Systems*, *42*(11), 1-13.

Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I. & Tuba, M. (2020). Monarch Butterfly Optimization Based Convolutional Neural Network Design, *Mathematics*, *8*(6), 1-32. Article ID 936.

Bezdan, T., Zivkovic, M., Tuba, E., Strumberger, I., Bacanin, N. & Tuba, M. (2020). Glioma Brain Tumor Grade Classification from MRI Using Convolutional Neural Networks Designed by Modified FA, *Advances in Intelligent Systems and Computing*: *Intelligent and Fuzzy Techniques: Smart and Innovative Solutions, 1197,* 955-963. Springer.

Boulent, J., Foucher, S., Théau, J. & St-Charles, P. L. (2019). Convolutional neural networks for the automatic identification of plant diseases, *Frontiers in Plant Science*, *10*: 941. DOI: 10.3389/fpls.2019.00941

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biological Cybernetics, 36,* 193–202.

Gao, Z., Li, Y., Yang, Y., Wang, X., Dong, N. & Chiang, H. D. (2020). A GPSO-optimized convolutional neural networks for EEG-based emotion recognition, *Neurocomputing*, *380*, 225-235.

Jia, P., Liu, Q. & Sun, Y. (2020). Detection and classification of astronomical targets with deep neural networks in wide-field small aperture telescopes, *The Astronomical Journal*, *159*(5), 212.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama S. & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia* (pp. 675-678).

Kebria, P. M., Khosravi, A., Salaken, S. M. & Nahavandi, S. (2019). Deep imitation learning for autonomous vehicles based on convolutional neural networks, *IEEE/CAA Journal of Automatica Sinica*, *7*(1), 82-95.

Krizhevsky, A. & Hinton, G. (2009). *Learning Multiple Layers of Features from Tiny Images*. Technical report, MIT and NYU.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks, *Advances in Neural Information Processing Systems*, *25*(2), 1097-1105. DOI: 10.1145/3065386

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. & Jackel, L. (1989). Handwritten Digit Recognition with a Back-Propagation Network, *Advances in Neural Information Processing Systems*, *2*, 396-404.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, *86(11),* (pp. 2278-2324).

LeCun, Y., Cortes, C. & Burges, C. J. (1998). *The MNIST Database of Handwritten Digits*. Available at: <<http://yann.lecun.com/exdb/mnist>>, last accessed: January 2022.

Lee, W. Y., Park, S. M. & Sim, K. B. (2018). Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm, *Optik*, *172*, 359-367.

Li, J. & Tan, Y. (2018). The bare bones fireworks algorithm: A minimalist global optimizer, *Applied Soft Computing*, *62*, 454-462.

Li, Y., Xiao, J., Chen, Y. & Jiao, L. (2019). Evolving Deep Convolutional Neural Networks by Quantum Behaved Particle Swarm Optimization with Binary Encoding for Image Classification, *Neurocomputing*, *362*, 156-165.

Liu, X., Wang, H., Zhang, X., Luan, H., Sha, Y. & Yan, Y. (2021). A Method Based on Multiple Population Genetic Algorithm to Select Hyper-Parameters of Industrial Intrusion Detection Classifier, *Studies in Informatics and Control, 30(3)*, 39-49. DOI: 10.24846/v30i3y202104

Pereira, S., Pinto, A., Alves, V. & Silva, C. A. (2016). Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images, *IEEE Transactions on Medical Imaging*, *35*(5), 1240-1251.

Rawat, W. & Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review, *Neural Computation*, *29*(9), 2352-2449.

Rosa, G., Papa, J., Marana, A., Scheirer, W. & Cox, D. (2015). Fine-Tuning Convolutional Neural Networks Using Harmony Search, *Lecture Notes in Computer Science: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, 9423*, 683-690. Springer, Cham.

Shen, W., Zhou, M., Yang, F., Yu, D., Dong, D., Yang, C., Zang, Y. & Tian, J. (2017). Multi-crop convolutional neural networks for lung nodule malignancy suspiciousness classification, *Pattern Recognition*, *61*, 663-673.

Shukla, S. K., Koley, E. & Ghosh, S. (2020). Grey wolf optimization-tuned convolutional neural network for transmission line protection with immunity against symmetrical and asymmetrical power swing, *Neural Computing and Applications*, *32*(4) 1-18. DOI: 10.1007/s00521-020-04938-z

Singh, P., Chaudhury, S. & Panigrahi, B. K. (2021). Hybrid MPSO-CNN: Multi-level Particle Swarm optimized hyperparameters of Convolutional Neural Network, *Swarm and Evolutionary Computation*, *63*: 100863. DOI: 10.1016/j.swevo.2021.100863

Sinha, T., Haidar, A. & Verma, B. (2018). Particle swarm optimization-based approach for finding optimal values of convolutional neural network parameters. In *2018 IEEE Congress on Evolutionary Computation (CEC),* (pp. 1-6). IEEE.

Strumberger, I., Tuba, E., Bacanin, N., Jovanovic, R., & Tuba, M. (2019a/b). Convolutional neural network architecture design by the tree growth algorithm framework. In *2019 International Joint Conference on Neural Networks (IJCNN)*, (pp. 1-8). IEEE.

Strumberger, I., Tuba, E., Bacanin, N., Zivkovic, M., Beko, M. & Tuba, M. (2019, May). Designing convolutional neural network architecture by the firefly algorithm. In *Proceedings of the 2019 International Young Engineers Forum (YEF-ECE),* Costa da Caparica, Portugal (pp. 59-65).

Tan, Y. & Zhu, Y. (2010). Fireworks algorithm for optimization, *Lecture Notes in Computer Science: Advances in Swarm Intelligence, 6145,* 355-364. Springer, Berlin, Heidelberg.

Wang, Y., Zhang, H. & Zhang, G. (2019). cPSO-CNN: An efficient PSO-based algorithm for fine-tuning hyperparameters of convolutional neural networks, *Swarm and Evolutionary Computation*, *49*, 114-123.

Zheng, Z., Yang, Y., Niu, X., Dai, H. N. & Zhou, Y. (2017). Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids, *IEEE Transactions on Industrial Informatics*, *14*(4), 1606-1615.

Zhu, W., Yeh, W., Chen, J., Chen, D., Li, A. & Lin, Y. (2019). Evolutionary convolutional neural networks using ABC. In *2019 Proceedings of the 11th International Conference on Machine Learning and Computing* (pp. 156-162).