Dropout Probability Estimation in Convolutional Neural Networks by the Enhanced Bat Algorithm

Nebojsa Bacanin
Faculty of Informatics and Computing
Singidunum University
Belgrade, Serbia
nbacanin@singidunum.ac.rs

Eva Tuba
Faculty of Informatics and Computing
Singidunum University
Belgrade, Serbia
etuba@ieee.org

Timea Bezdan
Faculty of Informatics and Computing
Singidunum University
Belgrade, Serbia
tbezdan@singidunum.ac.rs

Ivana Strumberger
Faculty of Informatics and Computing
Singidunum University
Belgrade, Serbia
istrumberger@singidunum.ac.rs

Raka Jovanovic
Qatar Env. and Energy Research Inst.
Hamad bin Khalifa University
Doha, Qatar
rjovanovic@qf.org.qa

Milan Tuba
Singidunum University
Belgrade, Serbia
tuba@ieee.org

***Abstract*—In recent years, deep learning has reached exceptional accomplishment in diverse applications, such as visual and speech recognition, natural language processing. The convolutional neural network represents a particular type of neural network commonly used for the task of digital image classification. A common issue in deep neural network models is the high variance problem, or also called over-fitting. Over-fitting occurs when the model fits well with the training data and fails to generalize on new data. To prevent over-fitting, several regularization methods can be used; one such powerful method is the dropout regularization. To find the optimal value of the dropout rate is a very time-consuming process; hence, we propose a model to find the optimal value by utilizing a metaheuristic algorithm instead of a manual search. In this paper, we propose a hybridized bat algorithm to find the optimal dropout probability rate in a convolutional neural network and compare the results to similar techniques. The experimental results show that the proposed hybrid method overperforms other metaheuristic techniques.**

***Index Terms*—convolutional neural network, dropout regularization, swarm intelligence, bat algorithm, hybridized bat algorithm**

I. Introduction

Deep learning has successful applications in various fields, such as natural language processing, speech processing, visual recognition. A convolutional neural network (CNN) [1] is a type of deep neural network used for different image processing tasks, such as object detection, image classification, pose estimation [2], scene labeling [3], face recognition [4], [5], etc. The CNN architecture consists of more layers and it mimics the visual cortex mechanism in the brain. Some of the famous architectures are LeNet [1], AlexNet [6], ZFNet [7], VGG [8], GoogleNet [9], ResNet [10], DensNet [11], SENet [12].

CNN structures have diverse alternatives in the literature; however, all of them have the same essential components. The essential components are the convolution layer, pooling layer, and fully-connected layer (FC-layer in short, or so-called dense layer). The convolution layer consists of the set of filters, and by performing convolution operation, the filters extract features from the input. On the convolved output, the activation function (transfer function) is applied. Sigmoid, tanh and ReLU [13] are the typically used activation functions. Each layer in the CNN takes an input from the previous layer. Filters in the CNN extracts information from the images; initial layers identify more general features such as edges, later layers discover more precise features, part of objects, then even later layers may detect complete objects, such as faces, or other complex patterns [14]. The filter size should be smaller than the size of the input; most commonly used filter sizes are 3×3, 5×5, and 7×7. The convolution operation on the input vector can be represented mathematically as follows:

$$z_{i,j,k}^{[l]} = w_k^{[l]} x_{i,j}^{[l]} + b_k^{[l]} \tag{1}$$

where $z_{i,j,k}^{[l]}$ is the output feature value of the k-th feature map (kernel) at i,j location. The input is denoted by x at i,j location, the filters are denoted by w, and b is the bias. The superscript l represents the l-th layer.

In the next step, the activation function is applied:

$$g_{i,j,k}^{[l]} = g(z_{i,j,k}^{[l]}) \tag{2}$$

where $g(\cdot)$ denotes a non-linear activation function, applied on the output $z_{i,j,k}^{[l]}$, which results with non-linear output.

The pooling layer is used between two convolution layers for reducing the resolution.

$$y_{i,j,k}^{[l]} = pooling(g_{i,j,k}^{[l]}) \tag{3}$$

The two most common types of pooling layers are *max* and *average* pooling.

The architecture, in the end, has one or more fully-connected layers, and the last one represents the output layer;

for example, in the case of the image classification task, typically softmax [15] is used.

A common issue in deep neural network models is the high variance problem (overfitting). Overfitting occurs when the model fits well with the training data and fails to generalize on the new, unseen data. In order to prevent overfitting, different regularization techniques can be used. Some of the effective regularization techniques are L_1 regularization, L_2 regularization (weight decay) [16], dropout [17], data augmentation, drop connect [18], early stopping and batch normalization (BN) [19] also has regularization effect.

In this work, we focus on the dropout regularization and optimizing its probability rate. This is an NP-hard optimization problem. Metaheuristic algorithms confirmed to be powerful in tackling such problems; therefore, we propose a metaheuristic approach to discover the optimal dropout rate. Metaheuristics are successfully applied to different real-life problems, some examples can be found [20], [21] [22], [23], [24], [25]. They have also proven to be successful in automatic CNN design [26], [27].

Metaheuristic algorithms are stochastic algorithms, and they have two major stages, namely exploration and exploitation (diversification and intensification). In the stage of exploration, the algorithm locally explores the search area; on the other hand, the exploitation process is responsible for the solution space investigation on a global scale. It is crucial to secure the right balance between these two processes. To enhance their performance, a number of hybridized versions were developed, some of them are [28], [29], [30], [31], [32], [33], [34]. Hybridized versions are developed by combining the advantages of one or more algorithms or part of the algorithms, which results in a synergistic effect.

The rest of this paper is organized as follows: Section II describes the dropout regularization. Section III describes the proposed method as well as the original bat algorithm. The experimental results are presented in Section IV, and the last section, Section V, concludes the paper.

II. Dropout Regularization

Dropout [17] is a regularization technique, which is used to prevent overfitting in neural networks. The idea behind dropout is to randomly drop hidden units from the network during the training, in case of convolutional neural networks, it only applies to last fully-connected layers, before the classification layer.

The feed-forward operation of a neural network can be described as:

$$z_i^{[l+1]} = w_i^{[l+1]} y^l + b_i^{[l+1]}, \tag{4}$$

$$y_i^{[l+1]} = g(z_i^{[l+1]}) \tag{5}$$

where the superscript l denotes the l-th hidden layer in the network, The input vector is denoted by z, the output vector is y. The weights and bias terms are denoted by w and b, respectively, and g is an activation function.

After applying the dropout regularization technique, the feed-forward operations can be described mathematically as follows:

$$r_j^{[l]} \sim Bernoulli(p) \tag{6}$$

$$\tilde{y}^{[l]} = r^{[l]} * y^{[l]}, \tag{7}$$

$$z_i^{[l+1]} = w_i^{[l+1]} \tilde{y}^l + b_i^{[l+1]}, \tag{8}$$

$$y_i^{[l+1]} = g(z_i^{[l+1]}) \tag{9}$$

where r represents a vector of independent Bernoulli random variables.

It is important to not use the dropout at test time, but only in the neural network training.

III. Proposed Method

This section describes the proposed method for the dropout probability selection in Convolutional Neural Networks. The learning process in CNN has four parameters:

1) the learning rate α,
2) penalty parameter (momentum) β,
3) weight decay λ,
4) dropout ratio p.

The purpose of this paper is to attain the optimal value of p; therefore, the 3-tuple (α, β, γ) is fixed, likewise in [35]. To find the optimal value of the dropout, we introduce a metaheuristic approach, the hybridized bat algorithm. The original Bat Algorithm (BA) [36] was proposed by Yang in 2010, and the procedure of the algorithm is explained in the next subsection, after that the hybridized algorithm is described. The hybridized algorithm has also been successfully applied to other real-life optimization problems [37].

A. Original Bat Algorithm

The bat algorithm simulates the echolocation behavior of bats. BA follows three simple rules:

1) Bats use echolocation to locate the prey and to assess the distance of an object and prey. When they broadcast noises, from the reflected echo, they can identify different objects.
2) Each bat, in the group, flies randomly while looking for the prey, the bats' position is x_i, they fly with velocity v_i, fixed frequency f_{min}, varying wavelength λ and loudness A_0. Depending on the target's closeness, the bats are able to adjust the pulse emission rate of r, which values are between 0 and 1.
3) For the sake of simplicity, the loudness ranges from a large positive value A_0 to a minimum constant A_{min}. The loudness parameter is used to control the exploration and exploitation process.

In the BA, positions of bats represent solutions. Movement of each bat is controlled by the following parameters: velocity, frequency, pulse rate, loudness nd wavelength. These parameters are used to define movements that ensure proper balance between exploration and exploitation. Initially, the algorithm generates N random solutions. The population of solutions is represented as a matrix X:

$$X_{1,1}, X_{1,2}, ..., X_{2,1}, X_{2,2}, ...X_{N-1,k-1} \tag{10}$$

If the objective function should be minimized, it can be described as follows:

$$\min\ f(x),\ x = (x_1, x_2, x_3, ..., x_j, x_D) \in S, \tag{11}$$

where x denotes a real vector with $D \geq 1$ parameters. $S \in R^D$ is a D-dimensional search space, which values are within the lower (lb) and upper bounds (ub).

$$lb_i \leq x_i \leq ub_i, \quad i \in [1, D], \tag{12}$$

The solutions are generated randomly by the following equation:

$$x_{i,j} = lb_j + \phi * (ub_j - lb_j), \tag{13}$$

where the j-the parameter of th i-th solution is represented by $x_{i,j}$. Parameter ϕ is a random number drawn from the uniform distribution. The lower and upper bound of the j-th parameter are denoted by ub_j and lb_j, respectively.

Similarly, the velocity is calculated by using the following formula:

$$v_{i,j} = lb_j + \phi * (ub_j - lb_j), \tag{14}$$

The position and the velocity of each solution, at time step t, are updated by using the following equations:

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

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

$$x_i^t = x_i^{t-1} + v_i^t, \tag{17}$$

where the frequency is denoted by f_i, the value of β is drawn from the uniform distribution. The global best solution is indicated by x_*. The velocity of i-th solution at iteration t is represented by v_i^t. The location of the bat i at iteration t is denoted by x_i^t.

The exploration process in the algorithm is executed by a random walk which can be specified mathematically as follows:

$$x_{new} = x_{old} + \epsilon A^t, \tag{18}$$

where A^t represents the mean value of all solutions' loudness and it is scaled by ϵ, which is a random number, generated between -1 and 1.

When a solution hits a promising area, the parameter that represents a pulse rate increases and, on the other hand, the loudness decreases, and it is mathematically defined as follows:

$$A_i^t = \alpha A_i^{t-1},\ r_i^t = r_i^0[1 - exp(-\gamma t)], \tag{19}$$

$$A_i^t \to 0,\ r_i^t \to r_i^0,\ \text{while } t \to \infty \tag{20}$$

where the loudness of i-th solution, at time step t is denoted by A_i^t, the pulse rate is expressed by r. α and γ are constant values.

The pseudocode of the BA is detailed in Algorithm 1. More details about the original BA can be found in [36].

Algorithm 1 BA pseudocode

```
Randomly initialize the population of $N$ solutions (bats) $x_i$, for $i = 0, \dots, N$
Initialize the algorithm' parameters for each solution: velocity ($v_i$), pulse emission rate ($r_i$) and loudness ($A_i$)
Define the frequency of pulse ($f_i$) for each solution $x_i$
Set the iteration counter ($t$) to 0.
while $t < MaxIter$ do
    for $i = 1$ to $N$ (all $N$ solutions in the population) do
        Update the frequency, velocity and location by utilizing Eq. (15), Eq. (16) and Eq. (17)
        if $rand > r_i$ then
            Find the best solution $x_{best}$
            Perform the exploration process by using Eq. (18)
        end if
        Generate new random solutions
        if ($rand < A_i$ and $f(x_i) < f(x_*)$ then
            The newly generated solution is accepted
            Reduce $A_i$ and increase $r_i$ by using Eq. (19)
        end if
    end for
    Find the best current solution $x_*$
end while
Return the best solution
```

B. Hybridized Bat Algorithm

The BA is well studied and used swarm intelligence algorithm. Various modified version can be found in the literature [38]–[40]. In [38] it was proposed to dynamically adjust the BA parameters by using fuzzy-logic while that work was extended by proposing interval 2-type fuzzy logic in [39]. Comparison between swarm intelligence algorithms that use interval 2-type fuzzy logic for a dynamic parameters adjustment was presented in [40]. Another modification of the BA was proposed in [41] where in order to make BA more efficient and to improve the performance, the original BA is hybridized with the ABC (artificial bee colony) algorithm [42]. The authors adopted the onlooker bee mechanism from the ABC algorithm. The aim of using this mechanism was to improve the exploration in the BA and to speed up the convergence process. In return, runtime will be reduced. Detail analysis of the hybridized BA algorithm can be found in [42].

The hybridized algorithm adopts the intensification process by the mechanism of onlooker bee from the ABC algorithm while keeping the BA exploitation mechanism too. These two mechanisms were performed alternately, in one iteration the BA exploitation was used while in the next iteration the ABC exploitation mechanism was performed.

Exploitation process in the ABC algorithm was implemented by using onlooker bees. Onlooker bees mechanism chooses a food source for exploitation in the next iteration with probability proportionate to the food source fitness [?].

The fitness function of each solution is assessed by:

$$F_i = \begin{cases} \frac{1}{f_i} & \text{if } f_i \geq 0 \\ 1 + |f_i| & \text{otherwise,} \end{cases} \quad (21)$$

where the value of the objective function is denoted by f_i.

The selection probability is performed by the following equation:

$$p_i = \frac{F_{avg}}{\sum_{i=1}^{m} F_i} \quad (22)$$

where the selection probability is denoted by p_i. The average of all fitness values is represented by F_{avg}.

The intensification process is performed by:

$$x_{i,j} = x_{i,j} + \phi \cdot (x_{i,j} - x_{k,j}), \quad (23)$$

where the j-the parameter of th i-th solution is represented by $x_{i,j}$, ϕ is a random number drawn from the uniform distribution and $x_{k,j}$ is j-th parameter of a neighbor bat k.

If the new solution has a better fitness value than the older one, then it replaces the old solution; contrarily, the old solution is kept.

If the number of time step t is even, then the onlooker search is conducted in the algorithm, if t is odd, the bat algorithm search is performed.

In this hybridized BA, the diversification process is not changed which means that is performed by the utilization of random walk defined by Eq. (18).

The pseudocode of the hybridized bat algorithm is described in the Algorithm 2.

IV. Experimental Setup and Results

The proposed method for dropout rate optimization is tested on two different image classification tasks, on the MNIST [43] and CIFAR-10 [44] datasets, sample pictures of both datasets are given in the Fig. 2 and Fig. 1. The achieved results are compared to the ones in [35], where three other swarm intelligence approaches were used on the equivalent datasets. To make a fair comparative analysis, in this study, we use a similar parameter configuration to [35]. The comparison is made with the following metaheuristics: particle swarm optimization (PSO) [45], firefly algorithm (FA) [46], cuckoo search algorithm (CS) [47], bat algorithm (BA) and the proposed approach, the hybridized enhanced bat algorithm (BA-OM).

The hybridized BA was implemented in Java SE 10 (18.3), and for the proposed method, Deeplearning4j library was utilized. The simulation tests were performed on NVIDIA GTX 1080 GPU and machine with Intel® CoreTM i7-8700K CPU, 32GB RAM, Windows 10 OS.

Algorithm 2 Hybridized BA pseudocode

```
Objective function $f(x) = (x_1, x_2, x_3, ...x_D)$
Randomly initialize the population $X_{i,j} (i = 1, 2, 3, ...N)(j = 1, 2, 3, ...D)$
Initialize the velocity $(v_i)$, pulse emission rate $(r_i)$ and loudness $(A_i)$
At position $x_i$, define the frequency of pulse $(f_i)$
Set the iteration counter $(t)$ to 0
while $t < MaxIter$ do
  for $i = 1$ to $N$ (all $N$ solutions in the population) do
    if $t$ is even then
      Perform bat search procedure using Equation (15), (16) and (17)
    else
      Apply the onlooker search procedure by using Equation (23)
    end if
    if $rand > r_i$ then
      Choose the best solution
      Perform the exploration process by using Equation (18)
    end if
    Generate new random solutions
    if $(rand < A_i$ and $f(x_i) < f(x_*)$ then
      The newly generated solution is accepted
      Reduce $A_i$ and increase $r_i$ by using Equation (19)
    end if
  end for
  Find the best current solution $x_*$
end while
Return the best solution
```

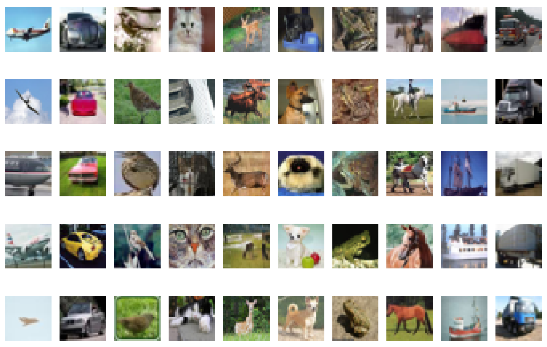
Fig. 1. CIFAR-10 sample pictures

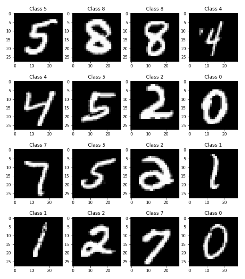
Fig. 2. MNIST sample pictures

1) Experimantal Setup: In this study, we used the same CNN architecture, like in [35]. For the CIFAR-10 dataset, a

CNN structure is utilized, which is deeper than the one used for MNIST. This network consists of three convolution layers, three pooling layers, and, at the end of the network, it has two FC-layer, the last FC-layer is the classifier (Figure 3). The network for MNIST dataset has two convolutional layers, two pooling layers, and one fully-connected layer; the last fully-connected layer is the classifier (Figure 4). Before the classification layer, the dropout regularization is employed.

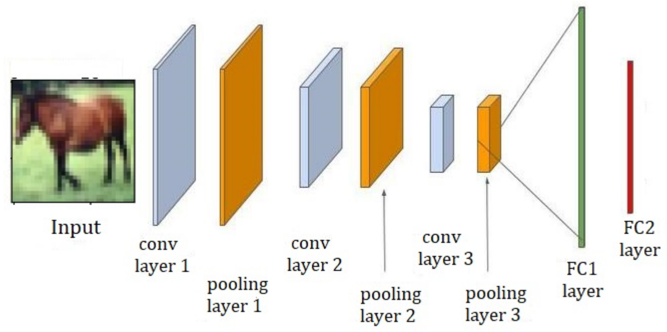

Fig. 3. CNN structure for the CIFAR-10 dataset

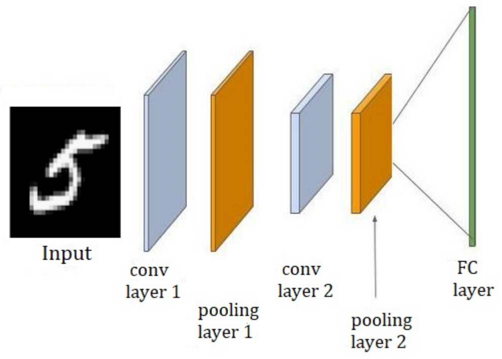

Fig. 4. CNN structure for the MNIST dataset

Both datasets are divided into training, validation and test set; the dataset configuration is presented in the Table I, where the number of images is specified and the batch size of each set; the CNN parameter configuration is shown in the Table II, and the metaheuristics control parameter setup is presented in the Table III.

TABLE I
CONFIGURATION OF THE DATASETS

Dataset	***Training set***	***Validation set***	***Test set***
CIFAR-10	20000 (100)	30000 (100)	10000 (100)
MNIST	20000 (64)	40000 (100)	10000 (100)

Since the dropout rate should be between 0 and 1, these two values are set up as lower and upper bound, respectively.

2) Experimental Results: In this paper we compared our proposed hybridized BA for the dropout rate optimization with

TABLE II
CONFIGURATION OF THE CNN PARAMETERS

Dataset	α	β	λ	p	$Iteration$
CIFAR-10	0.001	0.9	0.004	[0, 1]	4000
MNIST	0.01	0.9	0.0005	[0, 1]	10000

TABLE III
CONFIGURATION OF THE METAHEURISTICS' CONTROL PARAMETERS

Method	***Parameters***
PSO	$\omega = 0.7, c_1 = 1.7, c_2 = 1.7$
FA	$\alpha = 0.2, \beta_0 = 1.0, \gamma = 1.0$
CS	$\alpha = 0.8, \beta = 1.5, p = 0.25$
BA	$A = 0.5, rand = 0.5, f_{min} = 0, f_{max} = 2$
BA-OM	$A = 0.5, rand = 0.5, f_{min} = 0, f_{max} = 2$

the methods presented in [35]. The obtained results and the comparison analysis is presented in the Table IV and Table V, and the results are visualized in Fig. 5 and Fig. 6.

Based on the results presented in Table IV it can be concluded that the improved BA approach produces better results than other counterparts in dropout rate optimization for CIFAR-10 dataset. The accuracy was improved in comparison of the original BA and the proposed BA-OM method, 71.43% versus 71.76% which justifies the usage of the hybridized BA. The BA-OM algorithm resulted with 71.76% accuracy on the test set with dropout probability of $p = 0.6710$ in case of CIFAR-10 dataset while the second best method was the particle swarm optimization that achieved the accuracy of 71.55%. It should be mentioned that the main focus in this paper was optimizing the dropout rate while it is well-known that the success of the classification by the CNN depends also on various other parameters as well as on the chosen structure. To provide a fair comparison of the dropout rate optimization, all CNN's parameters and CNN structure were the same as in [35]. Due to a very specific research, the difference in the accuracy is not large. The third rated metaheuristic is the firefly algorithm that found dropout probability to be 0.6629 and the accuracy was 71.52%.

The accuracy on the MNIST dataset achieved by the proposed BA-OM is 99.19% with $p = 0.5216$ while the second rated algorithm is again the PSO that obtained the accuracy of 99.17% whith the $p = 0.4559$. Same as in the case with the CIFAR-10 dataset, the third rated metaheuristic is the firefly algorithm with the accuracy of 99.16%.

It should be noticed that, in practice, the commonly used default dropout rate is 0.5. Based on the presented results and by interpreting the them, it can be seen that the best accuracies are obtained when the dropout rate is greater than 0.5 and even 0.6 in the case of the CIFAR-10 dataset.

V. CONCLUSION

Overfitting is a common issue in deep neural networks. In order to prevent over-fitting, different regularization techniques

TABLE IV
CIFAR-10 ACCURACY ON THE TEST SET

Method	α	β	λ	p	$Accuracy(\%)$
PSO	0.001	0.9	0.004	0.6655	71.55%
FA	0.001	0.9	0.004	0.6629	71.52%
CS	0.001	0.9	0.004	0.6270	71.16%
BA	0.001	0.9	0.004	0.6430	71.43%
BA-OM	0.001	0.9	0.004	0.6710	71.76%

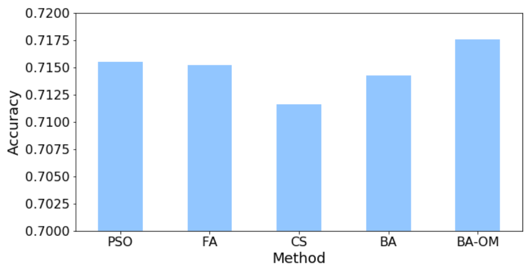

Fig. 5. CIFAR-10 accuracy comparison

TABLE V
MNIST ACCURACY ON THE TEST SET

Method	α	β	λ	p	$Accuracy(\%)$
PSO	0.01	0.9	0.0005	0.4559	99.17%
FA	0.01	0.9	0.0005	0.4630	99.16%
CS	0.01	0.9	0.0005	0.4883	99.14%
BA	0.01	0.9	0.0005	0.4988	99.13%
BA-OM	0.01	0.9	0.0005	0.5216	99.19%

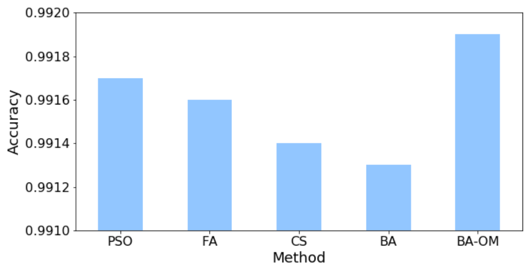

Fig. 6. MNIST accuracy comparison

can be used, one such powerful method is the dropout regularization, where the optimal probability should be determined. To find the optimal value by exhaustive search is a time-consuming process; therefore, in this research, we recommend the dropout probability fine-tuning by the hybridized bat algorithm. The obtained results are compared to similar metaheuristic techniques, PSO, BA, FA, and CS. The performance of the proposed method exceeds other approaches.

The approach is very promising, and in future work, we are going to investigate more algorithms for the same issue, as well as to implement it on other datasets.

REFERENCES

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[2] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, (Washington, DC, USA), pp. 1653–1660, IEEE Computer Society, 2014.

[3] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, pp. 1915–1929, Aug. 2013.

[4] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, June 2014.

[5] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[7] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *LNCS: Computer Vision, ECCV 2014 - 13th European Conference, Proceedings*, vol. 8689 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 818–833, Springer Verlag, 2014.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[9] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[11] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, July 2017.

[12] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[13] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[16] A. Y. Ng, "Feature selection, L1 vs. L2 regularization, and rotational invariance," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pp. 78–, ACM, 2004.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[18] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International conference on machine learning*, pp. 1058–1066, 2013.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37 of *Proceedings of Machine Learning Research*, pp. 448–456, 2015.

[20] I. Strumberger, M. Minovic, M. Tuba, and N. Bacanin, "Performance of elephant herding optimization and tree growth algorithm adapted for node localization in wireless sensor networks," *Sensors*, vol. 19, no. 11, p. 2515, 2019.

[21] N. Bacanin, E. Tuba, T. Bezdan, I. Strumberger, and M. Tuba, “Artificial flora optimization algorithm for task scheduling in cloud computing environment,” in *Intelligent Data Engineering and Automated Learning – IDEAL 2019* (H. Yin, D. Camacho, P. Tino, A. J. Tallón-Ballesteros, R. Menezes, and R. Allmendinger, eds.), (Cham), pp. 437–445, Springer International Publishing, 2019.

[22] N. Bacanin and M. Tuba, “Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint,” *The Scientific World Journal*, vol. 2014, 2014.

[23] E. Tuba, I. Strumberger, D. Zivkovic, N. Bacanin, and M. Tuba, “Mobile robot path planning by improved brain storm optimization algorithm,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2018.

[24] M. Tuba and N. Bacanin, “JPEG quantization tables selection by the firefly algorithm,” in *IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, pp. 153–158, IEEE, 2014.

[25] E. Tuba, I. Strumberger, N. Bacanin, and M. Tuba, “Optimal path planning in environments with static obstacles by harmony search algorithm,” in *Advances in Harmony Search, Soft Computing and Applications*, (Cham), pp. 186–193, Springer International Publishing, 2020.

[26] I. Strumberger, E. Tuba, N. Bacanin, R. Jovanovic, and M. Tuba, “Convolutional neural network architecture design by the tree growth algorithm framework,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.

[27] I. Strumberger, E. Tuba, N. Bacanin, M. Zivkovic, M. Beko, and M. Tuba, “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, 2019.

[28] I. Strumberger, N. Bacanin, M. Tuba, and E. Tuba, “Resource scheduling in cloud computing based on a hybridized whale optimization algorithm,” *Applied Sciences*, vol. 9, no. 22, p. 4893, 2019.

[29] M. Tuba and N. Bacanin, “Artificial bee colony algorithm hybridized with firefly algorithm for cardinality constrained mean-variance portfolio selection problem,” *Applied Mathematics & Information Sciences*, vol. 8, no. 6, p. 2831, 2014.

[30] M. Tuba *et al.*, “Hybridized particle swarm optimization for constrained problems,” in *Sinteza 2019-International Scientific Conference on Information Technology and Data Related Research*, pp. 17–25, Singidunum University, 2019.

[31] I. Strumberger, N. Bacanin, and M. Tuba, “Hybridized elephant herding optimization algorithm for constrained optimization,” in *International Conference on Health Information Science*, pp. 158–166, Springer, 2017.

[32] I. Strumberger, E. Tuba, N. Bacanin, M. Beko, and M. Tuba, “Modified and hybridized monarch butterfly algorithms for multi-objective optimization,” in *International Conference on Hybrid Intelligent Systems*, pp. 449–458, Springer, 2018.

[33] N. Bacanin, M. Tuba, and I. Strumberger, “RFID network planning by abc algorithm hybridized with heuristic for initial number and locations of readers,” in *2015 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim)*, pp. 39–44, IEEE, 2015.

[34] N. Bacanin and M. Tuba, “Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators,” *Studies in Informatics and Control*, vol. 21, pp. 137–146, June 2012.

[35] G. H. De Rosa, J. P. Papa, and X.-S. Yang, “Handling dropout probability estimation in convolution neural networks using meta-heuristics,” *Soft Computing*, vol. 22, no. 18, pp. 6147–6156, 2018.

[36] X.-S. Yang, “A new metaheuristic bat-inspired algorithm,” in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, (Berlin, Heidelberg), pp. 65–74, Springer Berlin Heidelberg, 2010.

[37] I. Strumberger, N. Bacanin, and M. Tuba, “Constrained portfolio optimization by hybridized bat algorithm,” in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pp. 83–88, IEEE, 2016.

[38] J. Pérez, F. Valdez, and O. Castillo, “Modification of the bat algorithm using fuzzy logic for dynamical parameter adaptation,” in *IEEE Congress on Evolutionary Computation (CEC)*, pp. 464–471, IEEE, 2015.

[39] J. Perez, F. Valdez, O. Castillo, P. Melin, C. Gonzalez, and G. Martinez, “Interval type-2 fuzzy logic for dynamic parameter adaptation in the bat algorithm,” *Soft Computing*, vol. 21, no. 3, pp. 667–685, 2017.

[40] F. Olivas, L. Amador-Angulo, J. Perez, C. Caraveo, F. Valdez, and O. Castillo, “Comparative study of type-2 fuzzy particle swarm, bee colony and bat algorithms in optimization of fuzzy controllers,” *Algorithms*, vol. 10, no. 3, p. 101, 2017.

[41] M. Tuba and N. Bacanin, “Hybridized bat algorithm for multi-objective radio frequency identification (RFID) network planning,” in *2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 499–506, May 2015.

[42] D. Karaboga and B. Akay, “A modified artificial bee colony (ABC) algorithm for constrained optimization problems,” *Applied Soft Computing*, vol. 11, no. 3, pp. 3021–3031, 2011.

[43] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” *http://yann.lecun.com/exdb/mnist/*, 2010.

[44] A. Krizhevsky, “Learning multiple layers of features from tiny images,” tech. rep., MIT and NYU, 2009.

[45] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4, Nov 1995.

[46] X.-S. Yang, “Firefly algorithms for multimodal optimization,” in *Stochastic Algorithms: Foundations and Applications*, pp. 169–178, Springer Berlin Heidelberg, 2009.

[47] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, “Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems,” *Engineering with computers*, vol. 29, no. 1, pp. 17–35, 2013.