Designing Convolutional Neural Network Architecture by the Firefly Algorithm

M SIVA RAJESWARI, B BASHU
KLR COLLEGE OF ENGINEERING & TECHNOLOGY

***Abstract*—** The firefly method framework is presented in this study for convolutional neural network architecture design. A specific subset of deep neural networks known as convolutional neural networks often consists of numerous convolutional, fully connected (dense), and pooling layers. Convolutional neural networks have been effective in solving a variety of image classification challenges and issues from the computer vision area. Finding the convolutional neural network design that achieves the highest performance for the particular application is one of the most difficult issues in this field. The hyper-parameter values of a network have a major impact on the overall network performance. This study focuses on convolutional neural network hyper-parameter optimization, which describes the design and organization of the network. The number of dense and convolutional layers, the number of kernels per layer, and the size of the kernel were the hyper-parameters considered in this study. We optimized the hyper-parameters using the well-known firefly method, a metaheuristic for swarm intelligence. Our suggested framework's performance, reliability, and quality were evaluated against the MNIST dataset. Empirical findings indicated that the suggested paradigm performs well in this area.

Index Terms—swarm intelligence, firefly algorithm, convolu- tional networks, deep learning

I. INTRODUCTION

There are several real-world uses for machine learning techniques. These techniques emphasize learning data representations rather than particular tasks. Machine learning approaches include deep learning, which makes use of deep neural networks, which have many hidden layers between the input and output layers. The only hidden layers in traditional artificial neural networks (ANNs) are one or two [1].

Deep learning has shown to be a reliable and powerful technique that can handle a variety of difficult machine learning jobs. Image recognition [2], voice recognition [3], and reinforcement learning [4] are a few examples.

Convolutional neural networks (CNNs), a specific subset of deep learning networks, first appeared in 1998 [5]. The fundamental concept underlying this strategy was to combine a trainable fully connected network classifier with a module for learning features [6]. CNNs are often implemented using a number of convolutional, dense (completely connected), and pooling layers. The majority of research on CNNs, according to a review of the literature, has been concentrated on training CNNs [7] and using them in real-world settings like computer vision [8]. In the current body of computer science literature, there are very few works that discuss CNNs hyper-parameter optimization [2], [9].

The CNNs' design is directly influenced by the CNNs' hyper-parameter optimization. The CNNs network design involves selecting values for a variety of hyper-parameters, such as the number of hidden layers and the number of units per hidden layer, the kernel size of a layer, the number of fully connected layers, the learning rate, the kind of activation function, the layer-ordering, etc. The model selection challenge in machine learning may be understood as the design of neural network architecture [2]. Despite several successful CNNs implementations for various real-world issues, designing CNNs architectures is still difficult since each particular challenge necessitates a unique CNNs's structure.

Three typical methods for choosing CNN's hyper-parameters are covered in the majority of literature sources: manual search, grid search, and random search [10]. A researcher must have extensive prior expertise in this field to conduct a manual search since, using this method, the researcher manually selects a set of CNN hyper-parameter values while being guided by his or her intuition. A CNN must be trained each time a researcher changes the values of a hyper-parameter, and CNN training is costly in terms of the use of computer resources. Reproducible results may be produced by using grid search, although this approach

is ineffective for high dimensional parameter space searches. Additionally, this approach wastes computer resources by investigating (in most instances) hyper-parameter values that have no bearing on the issue at hand.

Finally, the issue of undersampling crucial dimensions is avoided by random search, which is more effective than grid search [10]. The random search has the disadvantage of being non-adaptive since it does not take prior findings into account when choosing which hyper-parameters should be investigated. Because of this, combining manual and grid searches is the preferred course of action [10].

By looking at the accessible sources from the literature in this field, it can be deduced that there aren't many studies that have utilized genetic algorithms (GA) and swarm intelligence, two types of metaheuristics inspired by nature, to the process of CNN hyper-parameter optimization. Additionally, it should be highlighted that metaheuristic algorithms may provide superior outcomes than the methods mentioned above. On the other hand, a number of metaheuristic techniques were used to the backpropagation mechanism to substitute a stochastic gradient descent (SGD) [12, 13]. Swarm intelligence and metaheuristics inspired by nature

Swarm intelligence and evolutionary algorithms (EA) are the two main types of nature-inspired metaheuristics. Swarm intelligence algorithms imitate the collective behavior and social interactions between members of swarms, such as groups of bees, ants, birds, fish, fireflies, etc., whereas EA are inspired by the process of natural evolution [14].

[15]

One of the most well-known swarm intelligence algorithms, artificial bee colony (ABC), has been successfully used to solve a variety of optimization problems, including generic benchmark problems [16], as well as several real-world issues [17], [18]. Elephant herding optimization (EHO) metaheuristics, which were first presented in 2015, simulate elephant herds and have several uses for benchmark [19] and real-world issues [20], [21].

The revolutionary swarm intelligence technique known as monarch butterfly optimization (MBO) was initially proposed for benchmark issues in 2015 by Wang and Deb [22]. Despite this, there are already a lot of MBO implementations for real-world NP-hard issues [23]. Another swarm method that falls under the category of state-of-the-art technology is moth search (MS) [24]. The MS has several real-world applications in addition to applications for benchmark issues [25], [26], and [27].

The seeker optimization algorithm (SOA), the bat algorithm (BA), the cuckoo search (CS), the brain storm optimization (BSO), and the fireworks algorithm (FWA) are a few other swarm algorithms that, in addition to the ones mentioned above, all perform well when dealing with NP-hard optimization.

This paper's major objective is to propose an effective hyper-parameter selection method for CNNs that makes use of the Firefly algorithm (FA), one of the most well-known swarm intelligence techniques. The study that was done for the purpose of this article focuses on the optimization of CNNs' hyper-parameters, which include the number of convolutional and dense layers, the number of kernels per layer, and the size of the kernels. These hyper-parameters determine architecture and structure of a network. Other hyper-parameters, such as the learning rate, activation function, dropout, etc., were not taken into account because to the restrictions on the amount of computational power available for simulations.

The effectiveness of the proposed framework and the quality of the solution were verified using the well-known MNIST dataset for hand-written digits [32]. Similar methodologies were used for comparative analysis, and for better comparison, the same parameter configuration as in [33] was used. This essay has four parts in addition to the introduction. A formulation of the CNN hyper-parameter selection issue is offered in Section II. In Section III, specifics of the FA swarm metaheuristics were provided. Section IV discusses the adaptation of FA for CNNs hyper-parameter optimization, empirical findings, and performance validations of the suggested framework, while Section V wraps up this study and makes recommendations. future investigation in this area.

II. PROBLEM STATEMENT

In the research that is presented in this paper, a problemof finding satisfying CNNs network architecture, encoded asa set of hyper-parameters, is addressed. The following hyper- parameters were taken into account: number of convolutional layers, number of kernels per each layer and the layer size, and the number of fully-connected layers along with the size of each dense layer.

The CNN architecture is denoted as $h = \{h_{conv}, h_{fc}\}$, where h represents the set of the structural parameters con- sisting of sets h_{conv} and h_{fc}. Annotations h_{fc} and h_{conv} represent the parameters of the fully-connected and convolutional layers, respectively. The set of the convolutional layerparameters h_{conv} is defined as $h_{conv} = \{C_0, \ldots, C_{n-1}\}$, where n

represents the number of convolutional layers and $C_i = (k_{count}, k_{size})$ denotes the configuration tuple of the i-th layer - the number of kernels per i-th layer k_{count}, and the kernel size k_{size} of the i-th layer.

The $h_{fc} = s_0, s_{m-1}$ denotes the set of fully-connected layers, with m representing the number of fully-connected layers, and each layer i is defined by the layer size s_i.

With the symbol the set of possible CNNs network architectures (configurations) is denoted, and the objective of hyper-parameter selection problem is finding CNN archi- tecture h which minimizes error that depends on the specific task. Since this problem belongs to the group of NP hard optimization, with the goals of excluding some of the infeasible CNNs configurations from the search domain and decreasing the utilization of computing resources, we reduced the search space by introducing lower and upper boundaries of each hyper-parameter that was optimized.

III. Overview of the firefly algorithm

Firefly algorithm (FA) is well-known swarm intelligence metaheuristics. It was firstly proposed by Yang in 2008 [34], with the later improvements [35]. The FA metaheuristics has many successful applications for different kinds of tasks, for example, constrained benchmark problems [36], portfolio optimization [37], and wireless sensor networks localization [38]. Also, many hybridized versions of the FA may be found [18], [19].

Each solution in the population i in every iteration of algorithm's execution moves towards better solution j by using the following equation [39]:

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r_{i,j}^2}(x_j - x_i) + \alpha(\kappa - 0.5) \quad (6)$$

The FA metaheuristics models behavior of the group of fireflies. The basic assumption of this approach is that each firefly (solution) moves to the direction towards the more brighter (more attractive) firefly. Since the real firefly system is relatively sophisticated and complex, the following idealized rules were applied for the purpose of modeling this behavior and incorporating it into an optimization algorithm [34]: a) all fireflies in the population are unisex; b) the connection between attractiveness and the brightness is directly proportional;
c) the value of objective function determines the brightness.

where β_0 represents attractiveness at $r = 0$, α denotes the randomization parameter, κ is pseudo-random number in the range $[0, 1]$, and the distance between solutions i and j is denoted with $r_{i,j}$.

The concept of Cartesian distance is used for calculating the distance between solutions i and j [39]:

$$r_{i,j} = ||x_i - x_j|| = \sqrt{\sum_{k=1}^{N} (x_{i,k} - x_{j,k})}, \quad (7)$$

The brightness of a solution at a specific location x, in the case of maximization problems, can defined as $I(x) \propto f(x)$, where $I(x)$ denotes the attractiveness, while $f(x)$ represents the value of the objective function at this location. In the case of minimization problems, the following equation is utilized:

$$I(x) = \begin{cases} \frac{1}{1 + |f(x)|}, & if\ f(x) > 0 \\ 1 + |f(x)|, & otherwise \end{cases} \quad (1)$$

Generate initial population of fireflies x_i, $(i = 1, 2, 3, SN)$ Light intensity I_i at point x_i is defined by $f(x)$

With the increase of the light intensity and attractiveness, the distance from the source decreases, and vice-versa. This can be modeled with the following expression [39]:

$$I(r) = \frac{I_0}{1 + \gamma r^2}, \quad (2)$$

where the light intensity is represented as $I(r)$, the r is distance, and I_0 denotes the light intensity at the source. Moreover, the parameter γ models the air absorption.

In most FA implementations that can be found in the literature survey, the combined effect of both the inverse square law and absorption can be approximated using the following Gaussian form: Define light absorption coefficient γ Define number of iterations IN **while** $t < IN$ **do**

Algorithm 1 Original firefly algorithm

for i = 1 to SN **dofor** j = 1 to i **doif** $I_j < I_i$ **then**
Move firefly j towards firefly i in d dimension Attractiveness varies with distance r via exp[γr]Evaluate new solution, replace the worst withbetter solution and update light intensity –
end ifend for
end for
Rank all fireflies and find the current best
end while

$$I(r) = I_0 e^{-\gamma r} \qquad (3)$$

The attractiveness β varies with the distance $r_{i,j}$ between solutions i and j and it is directly proportional to the light intensity of the solutions [39]:

$$\beta(r) = \beta_0 e^{-\gamma r^2}, \qquad (4)$$

where β_0 is attractiveness at $r = 0$. Eq. (4) determines a characteristic distance $\Gamma = 1/\sqrt{\gamma}$ over which the attractiveness changes significantly from β_0 to $\beta_0 e^{-1}$.

To reduce the utilization of the computational resources (which is very important in the case of CNNs hyper-parameter optimization), in most practical implementations, the above expression is replaced by:

$$\beta(r) = \frac{\beta_0}{1 + \gamma r^2} \qquad (5)$$

The SN represents the total number of solutions in the population, IN is total number of algorithm's iterations, and t is the current iteration.

IV. Simulations, comparative analysis and discussion

At the beginning of the section, we first describe the adaptations of the FA for CNNs hyper-parameters selection problem along with the setup of algorithm's control parameters and the FA framework technology. After we show simulation results, comparative analysis and discussion.

A. Adaptation of the FA metaheuristics for CNNs hyper-parameters optimization

Each solution in the population is composed of the set of problem parameters that is subject to optimization. Like in very other swarm intelligence approach, the search process is conducting by executing two basic processes: exploitation (intensification) and exploration (diversification).

The process of exploitation of the FA metaheuristics isperformed by using Eq. (6). At the other side, the exploration is controlled with the values of parameters γ and β. There are two special implementations of the FA that are determined by the value of γ parameter [34]:

- if $\gamma = 0$, then $\beta = \beta_0$. In this case, value of the parameter β is the largest possible it could be, and solutions move towards other solutions in the largest possible steps.

the integer value only. For this purpose, after applying Eqs. (2) - (7), the results are rounded to the closest integer value. Taking into account that the lower and upper bounds of each parameter are relatively high, there was no need to adaptmetaheuristics for integer programming.

The fitness function that was employed is relatively straight-forward, and it is simply inversely proportional to the error on the test set for the dataset. Since, the objective is to minimize the classification error, to calculate the fitness for the i-thsolution, the following expression was utilized:f $\gamma =$, then $\beta = 0$. In this case, each solution moves in random steps meaning that the exploration is maximal,while practically there is no exploitation at all.

By adjusting the values of the parameters γ and β, the trade-of between exploration and exploitation is determined. In most practical implementations the value of the γ parameterbetween 0.01 and 100 is considered to be optimal, and empirical tests showed that an optimal value for the parameter β is 0.2, while for the α parameter is in the interval $[0, 1]$.

One of the greatest challenges in every metaheuristics implementation is how to encode potential problem's solution. In the FA approach for CNNs hyper-parameter selection, each solution is modeled with the set of hyper-parameters $h = h_{conv}, h_{fc}$. Thus, each solution represents one possible CNN architecture (configuration). To decrease the number of possible network configurations and to reduce the search domain, the convolutional layers are sorted in descended order, first with respect to the k_{size} and then based on the k_{count}. At the other side, the value of s, that denotes the layer size, was utilized for sorting fully-

connected layers. The population P in every time step t consists of n individuals, so that, $P^t = h_0, h_1, \ldots h_{n-1}$.

At the beginning of the FA's execution, in the phase of initialization, a population of uniformly distributed solutions is created by employing the following expression:

$$x_{i,j} = lb_j + \psi \cdot (ub_j - lb_j), \qquad (8)$$

where $x_{i,j}$ represents the j-th parameter of the i-th solution in the population, ψ is pseudo-random number in the range $[0, 1]$, and lb_j and ub_j are lower and upper bounds of the j-th parameter, respectively.

We imposed lower and upper bounds to the values of param- eters that are subject to optimization with the goal of shrinking the search space. For example, let the j-th parameter be the size of the convolutional layer. In the case of convolutional layer size (number of kernels), we set lb_j to 1 and ub_j to 128. In this way, guided by our previous experience, we excluded network configurations that don't make sense.

As already mentioned in the Section II, in this implemen- tation, we did not optimize hyper-parameters that may have real values, such are the dropout, the learning rate, etc., andin our model, each solution is composed of parameters with

where err_i is the classification error of the dataset for the i-th solution in the population.

By adapting the FA swarm intelligence metaheuristics for solving CNNs hyper-parameter optimization, we devised FA- CNN framework.

B. Parameter setup and dataset

For testing and validation purposes of the presented FA- CNN framework for CNNs hyper-parameter optimization problem, we used a well-known and widely used MNISTclassification dataset [32]. The MNIST dataset encompasses grayscale images of handwritten digits (from 0 to 9) withsize of $28x28$ pixels. The dataset uses 60,000 samples for training and 10,000 samples for testing the network, which yields in total 70,000 samples. To test our framework, we further divided the training set into 50,000 images for CNN training and 10,000 images for validation, similarly like in [33]. The pixel range of the images is scaled from $[0, 255]$ $Z$to $[0, 1]$ R.

To make a better distinction between various framework control parameters, we divided parameters into two groups: the control parameters of the FA metaheuristics and the CNN training parameters. Due to the fact that the process of training CNN is expensive in terms of computational resources, weset low values for the FA's control parameters as follows: the population size N was set to 8, while the maximum iteration number ($MaxIter$) was set to 20. Further, we set the value for the parameter γ to 1.0, and for parameters α and β to 0.5 and to 0.2, respectively. $\in$

The FA-CNN framework employs the Adam updater [40] with the learning rate $\vartheta = 0.0001$. The rectified linear unit (ReLU activation) function was used, with the batch size of 54, and for the cost function we chose mean squared error (MSE), similar as in [33]. Also, due to processor-intensive computations, each solution in the population was trained in 1 epoch.

The control parameters of the FA metaheuristics and the CNN training parameters are summarized in Table I.

As already stated in the Section II, in an optimization model, we included the following hyper-parameters: the number of convolution layers along with number of kernels (number of filters) per layer and kernel size, and the number of fully-connected (dense) layers along with the size of each

TABLE I
FRAMEWORK CONTROL PARAMETERS' ADJUSTMENTS

FA control parameters	
Parameter	**Value**
Population size (N)	8
Control parameter (λ)	1.0
Control parameter (α)	0.5
Control parameter (β)	0.2
Maximum iteration number ($MaxIter$)	20
CNN training parameters	
Parameter	**Value**
Updater (optimizer)	Adam
Learning rate (ϑ)	0.0001
Batch size	54
Training epoch per eval.	1
Cost function	MSE

fully-connected layer. To reduce the amount of parameters (weights), costs of computation, and to control overfitting, for each convolutional layer we used a pooling (downsampling) layer of the fixed size 2*x*2 with max pooling option. Also, with the goal of shrinking the search space and to exclude the most infeasible CNN configurations, we set lower and upper boundaries of values for each hyper-parameter that is addressed in optimization procedure. The hyper-parameters' boundaries, as well as its initialization are given in Table II.

TABLE II
HYPER-PARAMETERS BOUNDARIES AND INITIALIZATION

Parameter	Value	Parameter	Value
Number of convolutional layers	[0,6]	Initial convolutional layers	[1,2]
Number of fully-connected layer	[1,4]		

layer. The CNNs configurations generated by the FA-CNN framework are deeper than in [41], but significantly less complex as architectures found in [33]. Network architectures in the final population closely resembles the structure that was used in [11], where the authors proposed Multi-node Evolu- tionary Neural Networks for Deep Learning (MENNDL). The performance of the MENDL was evaluated on the CIFAR-10 dataset.

The solutions in the final (best) population were composed of the kernel size that ranges from 2*x*2 to 4*x*4, while the number of kernels per convolutional layer ranges from 42 to 128. As a comparison, the EA-based committee optimization (CEA-CNN) that was validated on the same dataset [33] generated solutions that consists of kernel sizes from 8*x*8 - 7*x*7 to 4*x*4-3*x*3, which is significantly larger than in the case of our proposed framework. Further, the solutions in the last generation of the FA-CNN consisted mostly of the dense layers with size from 40 to 154. One representative of the typical network structure that is found can be denoted as: $h_{conv} = (72, 3), (113, 2), (128, 4)$, and $h_{fc} = 101$.

The networks that showed best performance were created only after first 12 to 16 iterations in all conducted runs of the FA-CNN framework. After many empirical tests, we drew conclusion that the kernel size of the convolutional layers has much greater influence on the overall network performance than the number of filters (kernels). Parameter values of the top 5 performing network configurations, generated in the last population, are summarized in Table III.

TABLE III
BEST HYPER-PARAMETER VALUES IN THE LAST POPULATION

Parameter	Best 5 solutions
kernel 1	3
output 1	50-115
kernel 2	2-4
output 2	96-128
kernel 3	2-4
output 3	82-128
fc layer 1	50-148

Parameter	Best 5 solutions
Initial fully-connected layers	[1,2]
Kernel size	[1,8]
Number of kernels per conv. layer	[1,128]
Fully-connected layer size	[16,2048]

C. FA-CNN framework technology and infrastructure

We developed and implemented FA-CNN framework using deep learning programming library Deeplearning4j (URL: https://deeplearning4j.org/), that was written for the Java and Java Virtual Machine (JVM). The FA metaheuristics was developed in Java SE 10 (18.3) version.

To improve the execution speed of conducted experiments, due to the extensive utilization of computer resources, we adapted framework for the execution on the graphical process- ing unit (GPU) using NVIDIA's ™CUDA parallel computing platform and programming model. All tests were executed on the computer platform with 6 NVIDIA GTX 1080 GPUs each having 2560 ™CUDA cores.

D. Empirical results and comparative analysis

With the objective of generating real results and to measure performance in objective way, the algorithm was executed in 10 independent runs with different pseudo-random number seeds. The final population in most runs was composed mostly of solutions with only 3 convolutional layers and 1 dense

We further observed that the best hyper-parameter values ranges are similar as in [11]. Some of the network architectures

found during the FA's search process are shown in Figure 1. f We stated above that for each of 10 FA-CNN runs weused different pseudo-random number seed, and in everyrun, the algorithm has found networks with best performanceafter 12-16 iterations. Such behavior proves that the FA-CNNframework represents robust and reliable solution for CNNs hyper-parameters optimization.

In all conducted tests, the objective function that was subjectto optimization process was classification test error (expressedin percentage).

In order to objectively measure robustness and solutions' quality of the proposed FA-CNN framework for hyper- parameter optimization, we performed comparative analysis with other state-of-the-art metaheuristics that were validated on the same problem instance. Moreover, we utilized similar

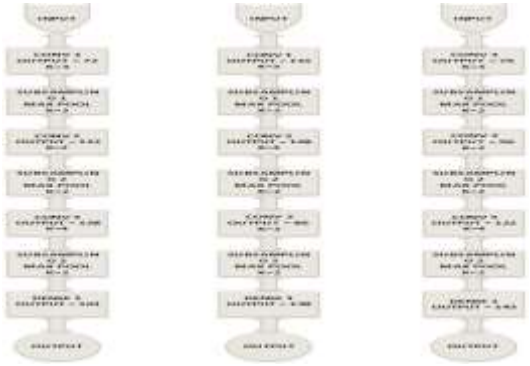

Fig. 1. Generated network architectures

parameters' values as in [33] to make comparisons more realistic.

The independent EA-based optimized CNN (IEA-CNN) and the joint EA-based committee optimization (CEA-CNN) with varying penalization *k* were tested in [33]. As shownin [33], the authors have found the best performing network configuration with test error of 0.24 % using CEA-CNN framework with *k* set to 2. At the other side, test error of only 0.23 % was produced by generated network configurations of our FA-CNN framework. We should also make observation that the FA-CNN framework utilizes less number of training epoch than approaches presented in [33]. As a consequence, FA-CNN framework is less computationally expensive.

We have also included other metaheuristics that were testedon the MNIST dataset under similar experimental conditionsin the comparative analysis. Comparative analysis is shown in Table IV. From the presented table it can be concluded thatthe FA-CNN framework performed better than other state-of- the-art approaches on this experimental instance, in term ofbest solutions' quality, as well as in term of utilization ofcomputational resources.

TABLE IV
BEST HYPER-PARAMETER VALUES IN THE LAST POPULATION

Method	Test error (%)
LeNet-5 [5]	0.95
Deeply Supervised Net [42]	0.39
Schallow CNN [41]	0.37
Recurrent CNN [43]	0.31
Gated Pooling CNN [44]	0.29
IEA-CNN [33]	0.34
CEA-CNN, k=1 [33]	0.26
CEA-CNN, k=3 [33]	0.28
CEA-CNN, k=2 [33]	0.24
TGA-CNN	**0.23**

V. CONCLUSION

The firefly method framework is presented in this study for choosing the best convolutional neural network design. The framework we developed is called FA-CNN framework. Finding the ideal hyper-parameter value set for a given application is one of the biggest and most significant issues in the field of convolutional neural networks.
The study done for this publication gives particular emphasis to the convolutional neural network hyper-parameters that control a network's design and structure. The number of dense and convolutional layers, the number of kernels in each layer, and the size of the kernels were taken into consideration. The limitations in computer capacity prevented the consideration of further factors.
The proposed FA-CNN system for handwritten digits was evaluated against the well-known MNIST dataset. The FA-CNN framework is a reliable, high-performing strategy for addressing this problem, according to comparison analysis with other meta-heuristics that were tried on the same problem instance and under comparable experimental settings.

ACKNOWLEDGMENT

This research is supported by Ministry of Education and Science of Republic of Serbia, Grant No. III-44006. This work was partially supported by Fundaçâo para a Ciência e a Tecnologia under Projects UID/MULTI/04111/0213 and UID/MULTI/04111/0216, UID/EEA/00066/2013 and foRESTER PCIF/SSI/0102/2017, and Grant IF/00325/2015.

REFERENCES

[1] Deep Learning, by I. Goodfellow, Y. Bengio, and A. Courville. 2016 MIT Press. http://www.deeplearningbook.org.
[2] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in GECCO '17 Proceedings, New York, NY, USA, pp. 497–504, ACM, 2017.
[3]A. Mohamed, G. Hinton, L. Deng, D. Yu, G. E. Dahl, N. Jaitly,

A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal Processing Magazine, vol. 29, no. 11, November 2012, pp. 82–97.

[4] Vnih, Kavukcuoglu, Silver, A. A. Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, and Ostrovski, J.

A. Sadik, I. Antonoglou, H. King, D. Kumaran, S. Petersen, C. Beattie,

"Human-level control through deep reinforcement learning," by D. Wierstra, S. Legg, and D. Hassabis, Nature, vol. 518, no. 2, pp. 529–533, February 2015.

[5] "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner.

[6] Y. Saez, A. Baldominos, and P. Isasi, "Evolutionary convolutional neural networks: An application to handwriting recognition," Neurocomputing, vol. 283, pp. 38–52, 2018.

[7] "Hybrid particle swarm training for convolution neural network (cnn)," in 2017 Tenth International Conference on Contemporary Computing (IC3), pp. 1-3, August 2017.

[8] "Convolutional neural networks for image classification," in 2018 International Conference on Advanced Systems and Electric Technologies, pp. 397–402, March 2018. N. Jmour, S. Zayen, and A. Abdelkrim.

[9] "Parameters optimization of deep learning models using particle swarm optimization," in 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1285–1290, June 2017. B. Qolomany, M. Maabreh, A. Al-Fuqaha, A. Gupta, and D. Benhaddou.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter opti- mization," J. Machine Learning Research, vol. 13, no. 2, pp. 281-305, February 2012.

[11] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in MLHPC '15 Proceedings, (New York, NY, USA), pp. 4:1–4:5, ACM, 2015.

[12] "Particle swarm optimization over back propagation neural network for length of stay prediction," Procedia Computer Science, vol. 46, pp. 268–275, 2015. A. Suresh, K. Harish, and N. Radhika. International Conference on Information and Communication Technologies, ICICT 2014, Bolgatty Palace and Island Resort, Kochi, India, 3–5 December 2014. Proceedings.

[13] "Particle swarm optimization for network-based data classification," Neural Networks, vol. 110, pp. 243–255, 2019. M. G. Carneiro, R. Cheng, L. Zhao, and Y. Jin.

[14] J. Del Ser, Z. W. Geem, and X.-S. Yang, "Foreword: New Theo- retical Insights and Practical Applications of Bio-Inspired Computation Approaches," 2019.

Genetic algorithms in search, optimization, and machine learning, by D. E. Goldberg. 1989, Longman Publishing Addison-Wesley.

[16] N. Bacanin and M. Tuba, "Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators," in Studies in Informatics and Control, vol. 21, no. 2, June 2012, pp. 137–146.

[17] "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, March 2015.

[18] N. Bacanin and M. Tuba, "Artificial bee colony algorithm hybridized with firefly metaheuristic for cardinality constrained mean-variance portfolio problem," Applied Mathematics & Information Sciences, vol. 8, no. 11, November 2014, pp. 2831–2844.

[19] I. Strumberger, N. Bacanin, and M. Tuba, "Hybridized elephant herding optimization algorithm for constrained optimization," in Hybrid Intelligent Systems (Cham), edited by A. Abraham, P. K. Muhuri, A. K. Muda, and N. Gandhi, pp. 158–166, Springer International Publishing, 2018.

[20] E. Tuba and Z. Stanimirovic, "Elephant herding optimization algorithm for support vector machine parameters tuning," in Electronics, Computers, and Artificial Intelligence (ECAI): Proceedings of the 2017 International Conference, pp. 1–5, June 2017.

[21] "Static drone placement by elephant herding optimization algorithm," Proceedings of the 24th Telecommunications Forum (TELFOR), November 2017. I. Strumberger, N. Bacanin, M. Beko, S. Tomic, and M. Tuba.

[22] "Monarch butterfly optimization," by G.-G. Wang, S. Deb, and Z.

May 2015, Neural Computing and Applications, pp. 1–20.

[23] "Monarch butterfly optimization algorithm for localization in wireless sensor" by I. Strumberger, E. Tuba, N. Bacanin, M. Beko, and M. Tuba

networks," in Radioelektronika, 2018's 28th International Conference, pp. 1-6, April 2018.

[24] G.-G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," in Memetic Computing, September 2016.

[25] G.-G. Wang, "The moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," Memetic Computing, vol. 10, no. 2, june 2018, pp. 151–164.

[26] "Moth search algorithm for drone placement problem," International Journal of Computers, vol. 3, April 2018. I. Strumberger, M. Sarac, D. Markovic, and N. Bacanin.

[27] "Wire-less sensor network localization problem by hybridized moth search algorithm," in 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), pp. 316-321, June 2018. I. Strumberger, E. Tuba, N. Bacanin, M. Beko, and M. Tuba.

[28] M. Tuba and N. Bacanin, "Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems," Neurocomputing, vol. 143, no. 2, 2014, pp. 197–207.

[29] Cuckoo Search and Bat Algorithm Applied to Training Feed-Forward Neural Networks by M. Tuba, A. Alihodzic, and N. Bacanin, pp. 139-

Springer International Publishing, 2015, p. 162.

[30] "Mobile robot path planning by improved brain storm optimization algorithm," 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1-8, July 2018. E. Tuba, I. Strumberger, D. Zivkovic, N. Bacanin, and M. Tuba.

[31] "Bare bones fireworks algorithm for capacitated p-median problem," E. Tuba, I. Strumberger, N. Bacanin, and M. Tuba, Advances in Swarm Intelligence (Y. Tan, Y. Shi, and Q. Tang, eds.), (Cham), pp. 283-291, Springer International Publishing, 2018.

[32] "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, Nov 1998, pp. 2278–2324. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner.

[33] "Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms," 2017 IEEE International Conference on Image Processing (ICIP), pp. 3924-3928, Sep. 2017. E. Bochinski, T. Senst, and T. Sikora.

[34] X.-S. Yang, Second Edition of Nature-Inspired Metaheuristic Algorithms. Summer 2010, Luniver Press.

[35] X.-S. Yang, "Firefly algorithm, stochastic test functions, and design optimisation," International Journal of Bio-Inspired Computation, vol. 2, no. 2, pp. 78–84, 2010.

[36] I. Strumberger, N. Bacanin, and M. Tuba, "Enhanced firefly algorithm for constrained numerical optimization, IEEE Congress on Evolutionary Computation," in Proceedings of the IEEE International Congress on Evolutionary Computation (CEC 2017), pp. 2120–2127, June 2017.

The Scientific World Journal, special issue Computational Intelligence and Metaheuristic Algorithms with Applications, vol. 2014, no. Article ID 721521, p. 16, 2014. N. Bacanin and M. Tuba, "Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint,"

[38] E. Tuba, M. Tuba, and M. Beko, "Two stage wireless sensor node localization using firefly algorithm," in Smart Trends in Systems, Security and Sustainability, edited by X.-S. Yang, A. K. Nagar, and A. Joshi, Springer Singapore, 2018, pp. 113–120.

[39] "Firefly algorithms for multimodal optimization," Stochastic Algorithms: Foundations and Applications, LNCS, vol. 5792, pp. 169–178, 2009. X.-S. Yang.

[40] J. Ba and D. P. Kingma, "Adam: A method for stochastic optimization.,"

2014, CoRR, vol. abs/1412.6980.

[41] M. D. McDonnell and T. Vladusich, "Enhanced image classification with a fast-learning shallow convolutional neural network," Proceedings of the International Joint Conference on Neural Networks (IJCNN), vol. 7, 2015, pp. 1–7.

[42] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets.," in AISTATS, edited by G. Lebanon and S. V. N. Vishwanathan, vol. 38 of JMLR

Proceedings, JMLR.org, 2015.

"Recurrent convolutional neural network for object recognition," by M. Liang and X. Hu, in 2015 IEEE Conference on Computer Vision and Pattern detection (CVPR), pp. 3367-3375, June 2015.

[44] "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in 2016 International Conference on Artificial Intelligence and Statistics, pp. 464472. C.-Y. Lee, P. W. Gallagher, and Z. Tu.