Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 3 (2024) 340 - 345 e-ISSN: 2580-0760

Metaheuristics Approach for Hyperparameter Tuning of Convolutional Neural Network

Hindriyanto Dwi Purnomo[1], Tad Gonsalves[2], Evangs Mailoa[3], Fian Yulio Santoso[4], Muhammad Rizky Pribadi[5]
[1,3,4,5]Department of Information Technology, Faculty of Information Technology, Satya Wacana Christian University, Salatiga, Indonesia
[2]Department of Information and Communication Sciences, Faculty of Science and Technology, Sophia University, Tokyo, Japan
[5]Department of Informatics, Universitas Multi Data Palembang, Palembang, Indonesia
[1]hindriyanto.purnomo@uksw.edu, [2]t-gonsal@sophia.ac.jp, [3]evangs.mailoa@uksw.edu, [4]fianyuliosantoso@gmail.com, [5]rizky@mdp.ac.id

Abstract

Deep learning is an artificial intelligence technique that has been used for various tasks. The performance of deep learning is determined by its hyperparameter, architecture as well as training (connection weight and bias). Finding the right combination of those aspects is very challenging. Convolution Neural Networks (CNN) is a deep learning method that is commonly used for image classification. It has many hyperparameters therefore tuning its hyperparameter is difficult. In this research, a metaheuristics approach is proposed to optimise the hyperparameter of convolution neural networks. Three metaheuristics methods are used in this research, ant colony optimization (ACO,) genetic algorithm (GA) and Harmony Search (HS). The metaheuristics methods are used to find the best combination of 8 hyperparameters with 8 options each which creates 1.6. 107 of solution space. The solution space is too big to explore using manual tuning. The Metaheuristics method will bring benefits in terms of finding solutions in the search space more effectively and efficiently. The performance of the metaheuristics methods is evaluated using MNIST datasets. The experiment results show that the accuracy of ACO, GA and HS are 99,7%, 97.7% and 89,9% respectively. The computational time for the ACO, GA and HS algorithms are 27.9 s, 22.3 s and 56.4 s respectively. It shows that ACO performs the best among the three algorithms in terms of accuracy however its computational time is slightly longer than GA. The experiment results reveal that the metaheuristic approach is promising for the hyperparameter tuning of CNN. Future research can be directed to solve larger problems or enhance the metaheuristics operator to improve its performance.

Keywords: *convolutional neural network; hyperparameter; metaheuristics; ACO; GA. HS*

How to Cite: H. Purnomo, Tad Gonsalves, Evangs Mailoa, F. J. Santoso, and M. R. Pribadi, “Metaheuristics Approach for Hyperparameter Tuning Of Convolutional Neural Network”, *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 8, no. 3, pp. 340 - 345, Jun. 2024
DOI: https://doi.org/10.29207/resti.v8i3.5730

1. Introduction

Deep learning is an artificial intelligence (AI) technique that is able to extract features and learn complex mapping. It can be used for large data sizes in which shallow techniques are insufficient to handle it. Deep learning is inspired by brain activity to learn high levels of feature hierarchy [1]. It learns by transforming raw feature space into another complex feature space [2]. Deep learning has been widely used such as for automated disease detection based on image data [3] [4], fake news detection [5], speech recognition [6], image classification [7] and image processing [8], [9].

Even though deep learning models have been used widely, the models still have limitations as they require high computational costs. Moreover, the performance of deep learning is significantly determined by its hyperparameter, weight and bias of its models as well as its architecture. Finding the right combination of those variables is not easy. Some approaches have been proposed to optimise those variables, such as the use of Grid Search, random Search and Bayesian Optimization for hyperparameters configure and the use of derivative methods to find the optimal values of the model weight and bias. The applicability of the methods is limited due to their high computational cost. The recent

Received: 24-02-2024 | Accepted: 23-05-2024 | Published Online: 01-06-2024

development of computational power enables researchers to explore new approaches to overcome its limitations.

Recently, new studies have been proposed to automate the search for deep learning design and parameters. One of the studies is neuro-evolution, the application of evolutionary computation to explore the huge search space of deep learning optimization problems. The use of an evolutionary algorithm enables the search process to obtain a near-optimal solution within an acceptable time. Examples of the study are the implementation of metaheuristics to enhance the CNN for characterization of abnormalities in breast images [10], and landslide susceptibility mapping [11], [12]. Some review studies also have been conducted to investigate the development of evolutionary algorithms in deep learning optimization. Tian and Fong [13] and Fong et.al [14] reviewed the implementation of a metaheuristic algorithm for training and parameters optimization of deep learning. Akay et al [2] provide a comprehensive review of the use of metaheuristics for optimizing deep learning models.

Although several research on metaheuristics algorithms for deep learning optimization have been proposed, there are several issues that still arise. First, deep learning models involve discrete and categorial parameters therefore derivation-based methods unsuitable for optimizing the problem. Second, deep learning models have high dimensions of hyperparameters that need to be tuned appropriately. Automatic hyperparameter optimization is needed to reduce the computational cost of running each different configuration. Third, finding the best deep-learning architecture for a specific problem is difficult. More research on the application of metaheuristics for deep learning optimization is needed and worth investigating in order to answer the issues.

In this research, we proposed a metaheuristics approach for hyperparameter tuning of Convolutional Neural Networks (CNN). CNN is widely used for processing and analyzing visual data such as video and images. The performance of CNN is significantly affected by the configuration of its hyperparameters, such as the number of convolutional layers, and the number of filters, along with the filter size, batch size, etc. For each problem, the CNN hyperparameter needs to be tuned as a CNN architecture will not generate satisfying results for all problems. Therefore, there is a need for a method to tune the CNN hyperparameters for each problem automatically. However, determining the right values of hyperparameters for a specific problem is not easy. Therefore, research on optimizing the hyperparameter of CNN is important. We believe that this research would be very beneficial for research in this domain. The rest of the paper is organized as follows: section 2 briefly describes the current research in this field, section 3 explains the proposed methods, section 4 presents the experimental result and discussion, and section 5 consists of the paper conclusion.

2. Research Methods

Deep learning has been used in various fields. The performance of deep learning is influenced by its training, hyperparameter configuration and deep learning architecture. Finding the best combination of deep learning hyperparameter configuration, training and architecture is a difficult task. Researchers have proposed methods to solve the problems and research in this domain is still ongoing.

A deep learning model that is widely used is the Convolutional Neural Network**.** It is primarily used for processing and analyzing visual data such as video and images. The model is developed by LeCun et all [15] to classify handwritten digits. The early model of CNN cannot perform well on more large-scale images because of a lack of training data and computer power. Several methods have been developed to overcome the difficulties in training CNN [16], [17]. The training will tune the CNN parameters (weights). Besides the parameter setting, the performance of CNN is highly dependent on its hyperparameter setting, such as the number of convolutional layers and filters, along with the filter size, batch size, etc. For each problem, the CNN hyperparameter needs to be tuned as a CNN architecture will not generate satisfying results for all problems. Therefore, there is a need method to tune the CNN hyperparameters for each problem. However, determining the right values of hyperparameters for a specific problem is not easy.

Optimizing the hyperparameter of CNN can viewed as an optimization problem. The problem belongs to an NP-hard problem, and it became a challenging task in the CNN domain. Some early methods for optimizing hyperparameters are random search [18], [19] and grid search [19], [20]. As the number of parameters increases, the use of random search and grid search for hyperparameters' tuning requires much time and knowledge from the domain. Therefore, there is a need for more sophisticated methods for optimizing the hyperparameter of a CNN. On the other hand, there is the metaheuristics method which is a modern optimization method that has been widely used for NP-hard problems.

However, there is only limited research that explores metaheuristics for hyperparameter tuning of CNN. In this paper, a metaheuristics approach is proposed for hyperparameter tuning of CNN. The performance of three metaheuristics methods, ant colony optimization genetic algorithm and harmony search are compared in this research. MNIST data set is used as the benchmark problem.

2.1 The Convolutional Neural Network Architecture

In this research, a hyperparameter optimization model is proposed for the convolutional neural network. A pre-trained model is used as the base model. The pre-trained model consists of the input layer, two convolutional layers, a dropout layer followed by max pooling, a flattened layer, batch normalization, two dense layers, a dropout layer and output. The base model is given in Figure 1.

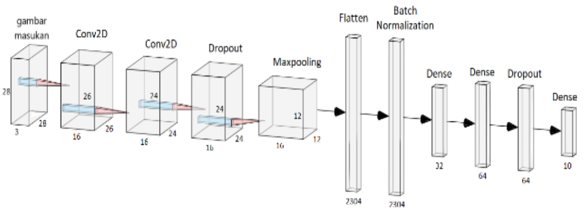


Figure 1. The Base model

The input for the CNN model is a 28 x 28 x 3 pixel. The input is forwarded to two sequential convolutional layers. The output of the second convolutional layer is forwarded to the dropout and maxpooling layers. The dropout layer is used to avoid overfitting [21]. The output of the max-pooling layer was flattened and then it was forwarded to the batch normalization layer. The batch normalization is used to improve the learning process [22]. The output of batch normalization is forwarded to dense layers and the dropout layer followed by the output layer.

Eight hyperparameters are optimized in this research. The hyperparameters are the number of neurons in the dense layers, dropout layers, the number of batches, the type of activation function, the type of optimizer and the type of loss function. Each hyperparameter has discrete values making the problem can be viewed as a combinatorial problem.

2.2 The Ant Colony Optimization

This section will focus on the description of the proposed ant colony optimization. An ant is a representation of a solution and is coded using an array as shown in Figure 2. Each cell is a representation of each hyperparameter. As each hyperparameter has its own option, therefore each hyperparameter is independent of each other.

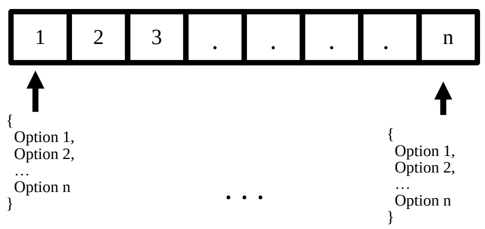


Figure 2. Solution representation

Solution generation is conducted by selecting an option for each hyperparameter based on the alternative option available. As each hyperparameter is independent of the other, a pheromone matrix is created for each hyperparameter. The matrix is given in Figure 3.

Figure 3. Pheromone matrix for each hyperparameter

The probability that an option is selected in a hyperparameter is given by Formula 1.

$$p_{i,k} = \frac{\tau_{,ki}}{\sum_{j \epsilon S} \tau_{j,k}}, \forall j \epsilon S_k \tag{1}$$

$p_{i,k}$ is the probability of option i for hyperparameter k, $\tau_{i,k}$ is the pheromone of option i for hyperparameter k, $S_k$is the set of option for hyperparameter k.

There are two operators for pheromone update, evaporation and reinforcement. Evaporation operator is used to simulate the natural phenomena when pheromone trails naturally evaporate over time. The pheromone update is is shown in Formula 2.

$$\tau_{i,k} = (1-\rho)\tau_{i,k} \tag{2}$$

ρ is evaporation rate. Reinforcement is a mechanism used to strengthen the pheromone trails on paths when an ant passes the trails. The reinforcement process aims to guide other artificial ants in the colony toward the most promising and high-quality solutions found during the optimization process. The reinforcement used in this research is shown in Formula 3.

$$\tau_{i,k} = \tau_{i,k} + 0.5 \tag{3}$$

The pseudocode for the proposed ant colony optimization is given as Pseudocode 1.

```
Pheromone initialization
   While stopping criteria is not met
        For each ant
           Generate a solution
           Pheromone update
                Evaporation
                Reinforcement
        End for
   End While
```

Each ant consists of a vector of hyperparameters. The hyperparameter vector is then used to configure a CNN. The performance of the CNN is evaluated using the *MNIST* datasets. The performance of the CNN is used to measure the performance of the ant (solution).

2.3 The Genetic Algorithm

This section describes the genetic algorithm used in this research. The solution representation is the same as used in ACO, shown in Figure 2. Each chromosome is a representation of a hyperparameter. The values of each chromosome are independent of each other. The work of the Genetic algorithm can be described in Figure 4.

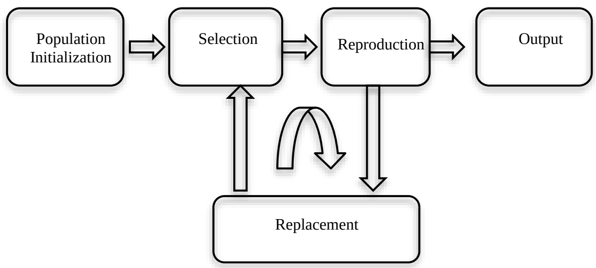


Figure 4. The Genetic Algorithm Framework

Population initialization is used to generate a set of initial individuals. An individual is a potential solution. The initial solution became the benchmark solution in the search process. Several individuals are selected from the population to generate new individuals. Roulette Wheel selection is used to select an individual from the population. The probability that an individual is selected is given as Formula 4.

$$p_x = \frac{f_x}{\sum_{l=1}^{l=M} f_l} \quad (4)$$

p_x is the probability of individual x is selected, f_x is the fitness value of individual x , M is the population size.

The fitness function used in this research is the accuracy of the CNN using the hyperparameter tuning given by the genetic algorithm. The accuracy is given as Formula 5.

$$f_x = \frac{TP+TN}{TP+FP+TN+FN} \quad (5)$$

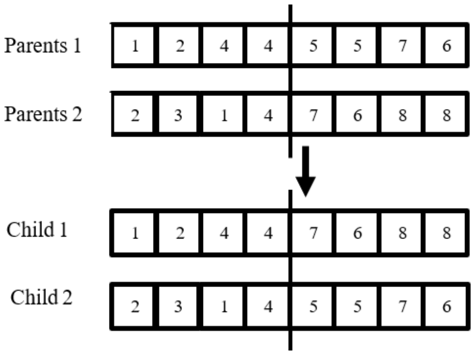


Figure 5. one point crossover

Individuals selected from the selection process are called parents. The parents are used to generate new individuals through a reproduction mechanism. Two operators are used in the reproduction mechanism, called crossover and mutation. Crossover is a method to generate a new individual by involving two or more parents. On the other hand, mutation is a method to generate a new individual involving only one parent. The crossover used in this research is a point crossover. In one point crossover, each parent is divided into two sections. A new individual is formed from the section combination of its parents. The one-point crossover is illustrated in Figure 5.

The mutation used in this research is a swap. The swap operator exchanges the values of a random point in a parent. The swap operator can be illustrated in Figure 6. The new individuals produced from the reproduction mechanism are then used to update the population. The update mechanism is called replacement or elitism. Only good individuals will be retained.

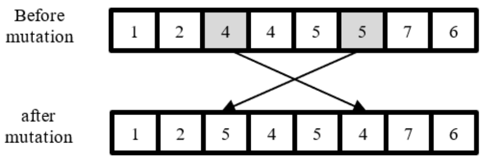


Figure 6. Swap mutation

The pseudocode of the genetic algorithm is given as Pseudocode 2. 2.

```
Generate initialization PopulationP(0)
   While stopping criteria is not met
        Fitness P(t)
        I(t) = Selection (P(t))
        If rand < probability of crossover
             A(t) = crossover (I(t))
             If rand < probability of mutation
                A(t) = mutation (A(t))
             Endif
        End if
        P(t) = replacement (P(t,A(t))
   End While
```

Each individual consists of a vector of hyperparameters. The hyperparameter vector is then used to configure a CNN. The performance of the CNN is evaluated using the *MNIST* datasets. The performance of the CNN is used to measure the performance of the individual (solution).

3.4 The Harmony Search

This section describes the harmony search algorithm used in this research. The solution representation is the same as used in ACO, shown in Figure 2. Each harmony is a representation of a vector of hyperparameters. In harmony search, there is a collection of solutions called harmony memory which can be represented in Figure 7.

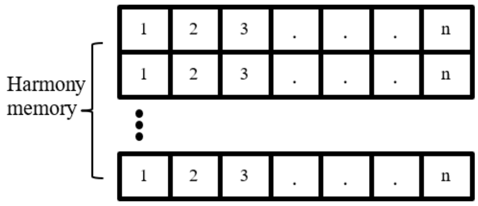


Figure 7. The representation of harmonious memory

The harmony search used in this research works as follows:

Generate harmony memory: in the first step, the harmony search will generate a set of initial solutions, called harmony memory. The solutions are then evaluated to determine their quality based on the objective function, shown in equation 5.

New harmony improvisation: there are three basic operations for harmony improvisation, random selection, harmony memory consideration (HMC) and pitch adjustment. Random selection means selecting a value of decision variable v_i randomly from an available value range of variable V_j. Harmony memory consideration means selecting a value of the decision variable from the harmony memory. The probability of choosing the element from the harmony memory is called the harmony memory consideration rate (HMCR). The probability of random selection is 1-HMCR. It can be formulated in Formula 6.

$$v_j' = \begin{cases} v_j \in \{e_j^1, e_j^2, \dots, e_j^{HMS}\} \text{ with prob } HMCR \\ v_j \in V_j \text{ with prob } 1 - HMCR \end{cases} \quad (6)$$

In the pitch adjustment, the obtained variable value from HMC is further adjusted to its neighbors. The probability of pitch adjustment is called pitch adjustment rate (PAR). It can be formulated in Formula 7.

$$v_j' = \begin{cases} v_j(k+m) & with\, prob\, HMCR\, x\, PAR \\ v_j' & with\, prob\, HMCR\, x\, (1-PAR) \end{cases} \quad (7)$$

$v_j(k)$ is the k^{th} element in V_j, and m is a neighboring index used for discrete variables (m = … , -2, -2, 1, 2, ...). In this research, m is either –1 or 1.

Solution update: after a vector of hyperparameter is determined, the CNN with the determined hyperparameter is trained with the data set. The performance of the CNN will be used to evaluate the quality of the harmony. When the new harmony has a higher quality than the worst harmony in the harmony memory, the new harmony is used to replace the worst harmony in the harmony memory.

3. Results and Discussions

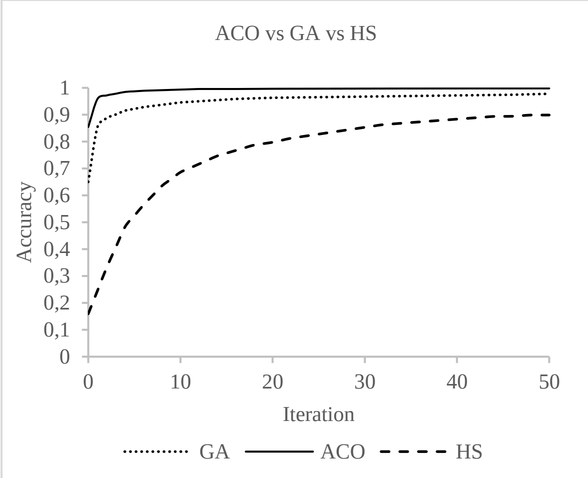
In this research, each hyperparameter has 8 values, making the search space 8^8 or 16.777.216. The values option for each hyperparameter is given Table 1.

Table 1. Values option for each hyperparameter

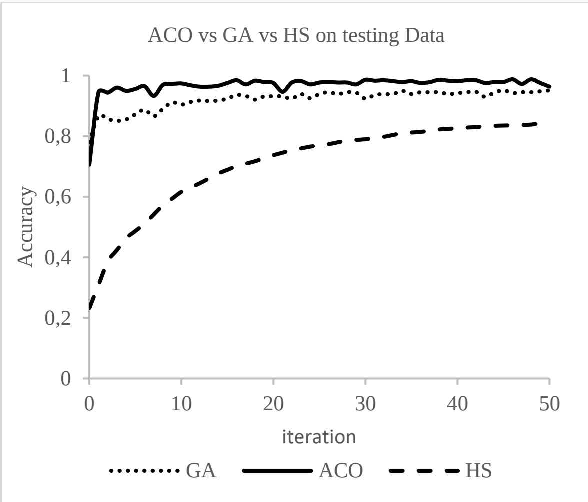
No	Hyperparameters	Options
1	The number of neurons in the first dense layer	16, 32, 64, 128, 256, 512, 1024, 2048
2	The number of neurons in the second dense layer	16, 32, 64, 128, 256, 512, 1024, 2048
3	The values of the first dropout layer	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
4	The values of the second dropout layer	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
5	The number of batches	16, 32, 64, 128, 256, 512, 1024, 2048
6	The type of activation function	relu, sigmoid, softplus, softsign, tanh, selu, gelu, linier
7	The type of optimizer algorithm	Adam, RMSprop, SGD, Adadelta, Adagrad, Adamax, Ftrl, Nadam
8	The type of loss function	Sparse Categorical Crossentropy, Categorical Crossentropy, Binary Crossentropy, Mean Absolute Error, Mean Squared Error, Squared Hinge, CategoricalHinge, cosine similarity

The number represents the value for each hyperparameter type as given in Table 1. The hyperparameter of the ant colony optimization is set as follows: the number of ants is 20, the evaporation rate is 0,25 and the reinforcement coefficient is 0.5. The performance of the ant colony optimization is compared to the genetic algorithm. The hyperparameter of the genetic algorithm is: the population size is 20, the crossover rate is 0.95 and the mutation rate is 0.1. The selection mechanism is Roulette Wheel Selection while one-point crossover and swap are used as the crossover and mutation respectively. The dataset used in this research is the *MNIST* dataset.

The results for the ACO, GA and HS are given in Figure 8. Using the training data, the accuracy of ACO, GA and HS are 99,7%, 97.7% and 89,9% respectively.



a. Accuracy in training data



b. Accuracy in testing data

Figure 8. The accuracy of ACO, GA and HS

The CPU time for the ACO, GA and HS are 27.9 s, 22.3 s and 56.4 s. The experiment shows that the metaheuristics approach work well to tune the hyperparameter of CNN. Based on the algorithms used in this research, the ACO perform the best compared to GA and HS, however, the CPU time is slightly longer than the GA. The HS perform the worst in terms of accuracy and CPU time.

4. Conclusions

A Metaheuristics approach is proposed for the hyperparameter optimization tuning of CNN. The methods used in this research are ant colony optimization, genetic algorithm, and harmony search. The methods have similar solution representations where a solution represents a set of options for each hyperparameter. Each hyperparameter is independent of each other. The experiment results show that the metaheuristics approach has good performance in tuning the hyperparameter of CNN. Ant Colony Optimization performs the best in terms of accuracy; however, its computation time is slightly longer than the Genetic Algorithm. The Harmony Search perform the least in terms of accuracy and computation time. Future research can be extended in several ways, such as finding the optimal CNN architecture for a given problem and enhancing the GA operator (crossover and mutation) to improve its performance.

Acknowledgements

This research was supported by Universitas Kristen Satya Wacana, Grand No. 062/SPK-PF/RIK/7/2023

References

[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets, Neural Computing,” *Neural Comput*, vol. 18, no. 7, pp. 1527–1554, 2006, doi: 10.1162/neco.2006.18.7.1527.

[2] B. Akay, D. Karaboga, and R. Akay, “A comprehensive survey on optimizing deep learning models by metaheuristics,” *Artif. Intell. Rev.*, vol. 55, pp. 829–894, 2021, doi: https://doi.org/10.1007/s10462-021-09992-0.

[3] S. Ahuja, B. K. Panigrahi, N. Dey, V. Rajinikanth, and T. K. Gandhi, “Deep transfer learning-based automated detection of COVID-19 from lung CT scan slices,” *Appl. Intell.*, vol. 51, pp. 572–585, 2020, doi: https://doi.org/10.1007/s10489-020-01826-w.

[4] M. A. Khan *et al.*, “Multiclass stomach diseases classification using deep learning features optimization,” *Comput. Mater. Contin.*, vol. 67, no. 3, pp. 3381–3399, 2021, doi: https://doi.org/10.32604/cmc.2021.014983.

[5] R. K. Kaliyar, A. Goswami, P. Narang, and S. Sinha, “FNDNet – A deep convolutional neural network for fake news detection,” *Cogn. Syst. Res.*, vol. 61, pp. 32–44, 2020, doi: https://doi.org/10.1016/j.cogsys.2019.12.005.

[6] Z.-T. Liu, M.-T. Han, B.-H. Wu, and A. Rehman, “Learning, Speech emotion recognition based on convolutional neural network with attention-based bidirectional long short-term memory network and multi-task,” *Appl. Acoust.*, vol. 202, 2023, doi: https://doi.org/10.1016/j.apacoust.2022.109178.

[7] E.-S. M. El-Kenawy *et al.*, “Advanced Meta-Heuristics, Convolutional Neural Networks, and Feature Selectors for Efficient COVID-19 X-Ray Chest Image Classification,” *IEEE Access*, vol. 9, pp. 36019–36037, doi: 10.1109/ACCESS.2021.3061058.

[8] A. M. Hafiz, R. A. Bhat, and M. Hassaballah, “Image classification using convolutional neural network tree ensembles,” *Multimed. Tools Appl.*, vol. 82, pp. 6867–6884, 2023.

[9] G. Chen, Q. Chen, S. Long, W. Zhu, Z. Yuan, and Y. Wu, “Quantum convolutional neural network for image classification,” *Pattern Anal. Appl.*, vol. 26, pp. 655–667, 2022.

[10] O. N. Oyelade and A. E. Ezugwu, “Characterization of abnormalities in breast cancer images using nature-inspired metaheuristic optimized convolutional neural networks model,” *Concurr. Comput. Pract. Exp.*, vol. 34, no. 4, p. e6629, 2021, doi: https://doi.org/10.1002/cpe.6629.

[11] W. L. Hakim *et al.*, “Convolutional neural network (CNN) with metaheuristic optimization algorithms for landslide susceptibility mapping in Icheon, South Korea,” *J. Environ. Manage.*, vol. 305, no. 1, p. 114367, 2022, doi: https://doi.org/10.1016/j.jenvman.2021.114367.

[12] Z. Chen and D. Song, “Modeling landslide susceptibility based on convolutional neural network coupling with metaheuristic optimization algorithms,” *Int. J. Digit. Earth*, vol. 16, no. 1, pp. 3384–3416, 2023, doi: https://doi.org/10.1080/17538947.2023.2249863.

[13] Z. Tian and S. Fong, “Survey of Meta-Heuristic Algorithms for Deep Learning Training,” in *Optimization Algorithms - Methods and Applications*, 2016, pp. 195–220. doi: 10.5772/63785.

[14] F. Simon, S. Deb, and X. Yang, “How Meta-heuristic Algorithms Contribute to Deep Learning in the Hype of Big Data Analytics,” in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, Rourkela, Odisha, India, 2017, pp. 3–25.

[15] Y. Le Cun *et al.*, “Handwritten digit recognition with a back-propagation network,” in *Advances in neural information processing systems 2*, 1990, pp. 396–404.

[16] O. Russakovsky *et al.*, “Imagenet large scale visual recognition challenge,” *Int. J. Confl. Violence*, vol. 115, no. 3, pp. 211–252, 2015, doi: https://doi.org/10.48550/arXiv.1409.0575.

[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proceedings of the International Conference on Learning Representations*, 2015, pp. 1–12. doi: https://doi.org/10.48550/arXiv.1409.1556.

[18] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, no. 10, pp. 281–305, 2012.

[19] P. B. Liashchynskyi and P. Liashchynskyi, “Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS,” *arxiv*, vol. @article{L, no. abs/1912.06059, 2019.

[20] E. Ndiaye, T. Le, O. Fercoq, J. Salmon, and I. Takeuchi, “Safe Grid Search with Optimal Complexity,” in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 4771–4780. doi: https://doi.org/10.48550/arXiv.1912.06059.

[21] M. Heidari, M. H. Moattar, and H. Ghaffari, “Forward propagation dropout in deep neural networks using Jensen–Shannon and random forest feature importance ranking,” *Neural Networks*, vol. 165, pp. 238–247, 2023, doi: https://doi.org/10.1016/j.neunet.2023.05.044.

[22] M. Segu, A. Tonioni, and F. Tombari, “Batch normalization embeddings for deep domain generalization,” *Pattern Recognit.*, vol. 135, p. 109115, 2023, doi: https://doi.org/10.1016/j.patcog.2022.109115.

